ZUBAIR SHAH MULTI-LAYERS FEATURE FUSION IN SSD FOR SMALL OBJECTS DETECTION NEU 2020

MULTI-LAYERS FEATURE FUSION IN SSD FOR SMALL OBJECTS DETECTION

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES

OF

NEAR EAST UNIVERSITY

By

ZUBAIR SHAH

In Partial Fulfillment of the Requirements for the Degree of Master of Science

in

Computer Engineering

NICOSIA,2020

MULTI-LAYERS FEATURE FUSION IN SSD FOR SMALL OBJECTS DETECTION

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By

ZUBAIR SHAH

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

NICOSIA,2020

ZUBAIR SHAH : MULTI-LAYERS FEATURE FUSION IN SSD FOR SMALL OBJECTS DETECTION

Approval of Director of Graduate School of

Applied Sciences

[illegible]

Prof.Dr.Nadire CAVUS

We certify this thesis is satifactory for the awarded of degree of Master of Sciences in Computer Engineering

Prof.Dr. Rahib ABIYEV [illegible]	Committe Chairman, DepartmentofComputer Engineering,NEU
Assoc. Prof. Dr. Kamil DIMILILER [illegible]	Automotive Engineer, Department of Computer Engineering, NEU
Assist.Prof.Dr. Elbrus IMANOV [illegible]	Supervisor, Department of Computer Engineering,NEU

I certify that this research work entitled “Multi-layers Feature fusion in SSD for Small Objects Detection” is my own work. No portion of the work presented in this research report has been submitted in support of another award or qualification either at this institution or elsewhere. Where material has been used from other sources it has been properly acknowledged / referred. If any part of this project is proved to be copied or found to be a report of some other, I will stand by the consequences.

Name, Last Name: Zubair Shah

Signature: [illegible]

Date: 28/12/2020

ACKNOWLEDGMENTS

I am deeply grateful to my supervisor **Asst.** Prof. Dr. ELBRUS IMANOV, for his guidance, support and patience. He has been invaluable source of knowledge and has certainly helped inspire many of the ideas expressed in this Thesis.

I would wish to express my sincere regards to the Chairman Department of Computer Engineering Prof. Dr. Rahib ABIYEV for his open hearted encouragement and good wishes

Our words will fail to express our deepest heartfelt thanks to our families, especially my parents, and my dear elder’s Brother all what they did, and still doing to help me be at this position and for their continuous support and encouragement.

ABSTRACT

The most useful and popular object identification method is SSD (single shot multi-box detection). At present, object detection using convolutional neural networks occupies a dominant position. However, the structure of the competent neural network faces hereditary difficulties: high-level networks have large receptive areas and meaning information has high capacity to illustrate, but the solution is diminished, the calculations are weak. In grassroots networks there are relatively small acceptable areas, and it has strong geometric detail information representation capability. However, the higher the resolution, the more information the capacity would have been so weak. SSD object prediction the multi-level feature uses mAP, predicting things with large fields of reception and small fields of prediction of things, It also uses high-level facilities to do. It makes difficulties: When using information about lower-level network functions to predict small things, due to the lack of standard high-level functions, the identification of small SSD objects is seriously compromised.

Based on the analysis and introduction of classic SSD algorithms, in this thesis, we aim to detect small objects at a fast speed, through present an approach which adapts the single socket multi-box (SSD) detector in relation to trading with precision and speed. An MSSD procedure of multistage components combination is offered to supply old knowledge in SSD, in order to improve the accuracy for small objects. In detail, the merge operation consists of two resource merge modules, a concatenation module and a sum of elements module, different in the way of adding contextual information. The VGGl6 and deep residual networks are used by the MSSD training to optimize candidate box regression and classification task input feature mAPs to improve detection accuracy and detection speed. With Residual network, this thesis uses the FPN-based network architecture to integrate high and low layers and improves the traditionally sampled structure. The high-level semantic information is integrated into the low-level network feature information, and the multi-scale feature mAPs for predicting the regression location box and the classification task input are enriched to improve the detection accuracy. Experiments are performed on the logo and VOC2007/2012 datasets which contains a large amount of small objects (objects of 50 pixels or less). Experimental results show that these two fusion modules obtain better mAP on PASCAL VOC2007 and Logo datasets than base line SSD, especially on some small objects categories.

KEY WORDS: Small object detection, single shot multi-box detector, MSSD feature fusion, Feature Pyramid Networks, real-time.

ÖZET

En kullanışlı ve popüler nesne tanımlama yöntemi SSD'dir (tek seferde çoklu kutu algılama). Şu anda, evrişimli sinir ağlarını kullanan nesne algılama baskın bir konumdadır. Bununla birlikte, yetkin sinir ağının yapısı kalıtsal zorluklarla karşı karşıyadır: yüksek seviyeli ağların geniş alıcı alanları vardır ve bu, bilginin gösterme kapasitesinin yüksek olduğu anlamına gelir, ancak çözüm azalmıştır, hesaplamalar zayıftır. Taban ağlarında nispeten küçük kabul edilebilir alanlar vardır ve güçlü geometrik ayrıntı bilgi gösterimi kapasitesine sahiptir. Bununla birlikte, çözünürlük ne kadar yüksek olursa, kapasite o kadar zayıf olurdu. SSD nesne tahmini, çok seviyeli özellik mAP kullanır, geniş alım alanları ve küçük tahmin alanları olan şeyleri tahmin eder, ayrıca yapmak için üst düzey olanaklar kullanır. Zorluklar yaratır: Küçük şeyleri tahmin etmek için alt düzey ağ işlevleri hakkındaki bilgileri kullanırken, standart üst düzey işlevlerin bulunmaması nedeniyle, küçük SSD nesnelerinin tanımlanması ciddi şekilde tehlikeye girer.

Klasik SSD algoritmalarının analizine ve tanıtımına dayanarak, bu tezde, tek soketli çoklu kutu (SSD) dedektörünü hassas ve hızlı ticarete göre uyarlayan bir yaklaşım sunarak, küçük nesneleri hızlı bir hızda tespit etmeyi hedefliyoruz. . Küçük nesnelerin doğruluğunu artırmak için SSD'deki eski bilgileri sağlamak için çok aşamalı bileşen kombinasyonunun bir MSSD prosedürü sunulmaktadır. Ayrıntılı olarak, birleştirme işlemi, bağlamsal bilgi ekleme şeklinde farklı olan iki kaynak birleştirme modülünden, bir birleştirme modülünden ve bir öğe toplamı modülünden oluşur. VGGl6 ve derin artık ağlar, MSSD eğitimi tarafından, algılama doğruluğunu ve algılama hızını iyileştirmek için aday kutu regresyonunu ve sınıflandırma görevi girdi özelliği haritalarını optimize etmek için kullanılır. Artık ağ ile bu makale, yüksek ve düşük katmanları entegre etmek için FPN tabanlı ağ mimarisini kullanır ve geleneksel olarak örneklenen yapıyı iyileştirir. Yüksek seviyeli anlamsal bilgi, düşük seviyeli ağ özelliği bilgisine entegre edilmiştir ve regresyon konum kutusunu tahmin etmek için çok ölçekli özellik mAP'leri ve sınıflandırma görevi girdisi, algılama doğruluğunu iyileştirmek için zenginleştirilmiştir. Logo ve büyük miktarda küçük nesneler (50 piksel veya daha küçük nesneler) içeren VOC2007 / 2012 veri kümeleri üzerinde deneyler yapılır. Deneysel sonuçlar, bu iki füzyon modülünün, özellikle bazı küçük nesne kategorilerinde, PASCAL VOC2007 ve Logo veri setlerinde temel SSD'ye göre daha iyi haritalama elde ettiğini göstermektedir.

ANAHTAR KELİMELER*:* Küçük nesne algılama, tek atışlı çoklu kutu dedektörü, MSSD özellik füzyonu, Özellikli Piramit Ağları, gerçek zamanlı.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	CONVOLUTIONAL NEURAL NETWORK
SSD	SINGLE SHORT MULTI-BOX DETECTOR
MSSD	MODIFIED SINGLE SHORT MULTI-BOX DETECTOR

CHAPTER 1

INTRODUCTION

1.1 Introduction

Fast growing also years with computer hardware and software from artificial intelligence, unmanned driving, intelligent transportation, travel, with identification and reuse of many other applications there is no doubt a need to include the goods. Computers should be used to detect and identify Real-time tracking, rapid tracking which makes it important as a fundamental, long-standing and difficult subject in computer vision, the discovery of object has always been for decades, athough it dominates an important and active area of research has been. Detecting and locating whether or not there is instance of object in a given image from given categories is the objective of object detection and return with the spatial position through a bounding box.

Recently, the techniques of deep learning [26, 27] have emerged as efficient and powerful methods for automatically learning features representation from data. Specifically, these techniques had already provided significant improvement as never before for object detection, which is a question that has mobilized large number of attention recently. Although difficulties of objects detection has seen a qualitative leap in scientific through a lot of research in natural scenes, it is yet far from being solved, especially for small objects. This problem is extremely relevant in many of today's challenging research applications like detecting traffic signs, pedestrians and cars on roads and a so on. Although CNNs have been proved efficient and effective on object detection, the reliable and accurate detection for small objects, due to their limited information and resolution in images deemed a quite challenging task [36],[35]which the small object activations become forwardly smaller with each pooling layer, as passing an image throughout a standard CNN architecture like ResNet or VGG16, and Current methods often cannot detect small size objects effectively as well as they have done for the large objects [10]. The challenges involved in the detection of small objects are multiple fold, but the biggest challenge stems from the comparatively small size of an object compared to its background in an image, e.g. the small object of interest occupies only 1 to 5 percent of pixels in an image. In addition, the input size for all these networks is indicated by in the place of storage over GPUs due to the enormous network running memory requirements. For instance, an SSD detection model [34] based on VGG16 [25] needs more than 10 gigabytes to handle only a single image with a 2048 R 2048 input size. Simplifying the network, e.g. using a shallow one, with a tradeoff of performance degradation is the only way to overcome the above mentioned problem. A second alternative solution in order to suits the memory is by down-sample the original image. However, It will be even more difficult to detect small objects.

This research work is based on existing SSD methods, on additional research and references, using resource maps at various scales of different layers to predict or anticipate objects, large predicted objects using more receptive large fields with high level resource information, though smaller. Receptive fields and low-level information about the characteristics used to predict small objects. When a person is at the spiritual level smaller than SSD detection if it is deficient may have adverse effects on objects. Solve this problem with top level iconic detail and little features facts must be combined. This research is high-level and low-level uses an FNF-Based network structure for feature fusion and expands the traditional ultra-sample structure First class with this faculty network information combines grammatical facts and predictive position of different field scales enriches the map on view and classification functions enter. To improve detection accuracy after extracting the resource maps at various scales, this thesis provides a forecast module consisting of residual units, extracts the deepest resources further and finally inserts them into the cash regression task and the classification task. Specialized information from this faculty network integrates with first-class grammatical fact and strengthens the outline map. Solved object detection problems accuracy and increased speed classification works. It has important practicability in artificial intelligence, unmanned driving, intelligent transportation, face recognition and other fields.

1.2 Evolution of Image Recognition

In 2012, object method ranked resources adopted the perception of maximization. First, someone item categories it is necessary to specify a particular feature so that it can be display correctly. The object can be completely represented by resource vectors after extracting sufficient resources from the training data, which will be used during the training time to be used during the test time to perform detection tasks and also to train a classifier. Drawback stand out, as the clarity of the function is very tricky also difficult get expand the model by adding an additional object to the detection list. In addition, the detection accuracy is not satisfactory In 2012, during the mass visual recognition challenge (ILSVRC,2012), Krysovsky's CNN model to classify images with extend training images in addition to GPU-CNN beat the high potential self-learning capabilities taking advantage of, CNN models are more appropriate and adaptable so that if appropriate training data is available, it will be used in other categories too many can be done. From that day on, CNN has become the primary tool for classifying objects. From 2012 to 2015, Research subsequently wrote regression heads for all CNNs currently in circulation, including VGG-Net [25], Overfeat-Net [24] and ResNet [14], thereby lowering the rate. 34% to 9% of individual object location error. Obviously, in the role of location and classification, CNN-based approaches performed very well.

The researchers also started working on various object detection tasks. In late 2014, several proposals for the CNN-based region and classification methods were developed [6] [7] [8] [21]. Essential i mean workplace proposition approaches create applicant zone which are likely to include objects and then classify all those regions later in practical terms, these regional proposition and classification methods could achieve very precise precision. High [24] [25] [14]. However, the part of the region's proposal really takes a long time, slowing down the entire system.

1.3 Motivation

Although the natural landscape as objects research has long been the subject of research and it is a computer big challenge has been made in sight, small the question of finding objects has been neglected for a long time of course, the object detector is large and will work well on medium sized objects, but images due to the "low resolution" size of little objects, these detectors are the objects perform poorly for the general function of identification and detection identification to exhibit. At present, the main factors restricting the development of object detection include the following aspects:

- Image half-and-half analysis of a comprehensive iterative neural network: the multi-function function of the context function, the virtual neural network is vulnerable to structural problems. High-level networks have enormous acceptable fields and powerful ability to present meaningful information, but low resolution is weak

• geometric detail information representation capabilities. Yielding area for low-rate network is comparatively low, and Strength to represent geometric details is strong. Despite the high resolution, the capacity to represent semantic information is low. Multi-scale object positioning, objects are predicted by multi-scale feature maps, large objects are predicted by feature information of high-level and small objects are predicted by feature information of low-level with larger receptive fields. This raises concerns when information from the purpose of poor system network is used portend small ones, the absence of next level voice functions indicates poor ones.

• Hard to imagine real-time generation of high-resolution images or videos: A object with a simple background, sufficient light, no obstruction, and a shooting angle of front is relatively easy to detect, because this kind of image is the easiest type in object detection. Many methods can achieve 100% accuracy for this kind of image. The detection rate is greatly reduced when the background is mixed with the object, there are occludes near the object, the light intensity is too weak, and the object pose changes. Considering the real time detection, detecting and recognizing small objects accurately in a fast way is hardly achieved by these studies. A large improvement ion speed has presented by The Single Shot Multi box Detector (SSD [10]) fast detector. Thus, currently, the precision / speed compromise is still difficult to balance simultaneously and remains a major problem for future research to be solved. We are inspired here by

• Developed a latest idea "MSSD technique for the sake of related information to SSD libraries". Thus, today, the compromise between precision and speed is also still difficult to balance simultaneously; it remains largely a vital problem for future research sorted out.

1.4 Contribution

The main purpose of the current topic is to quickly detect small things. Our partnership includes the following parts to be included documents. Data set up pre-processing for 8 classes to identify training, verification and data needed to identify something

• We check if the object detector is SSD because the speed and speed trade-off original architecture for normal small object is good example in term of accuracy. Various datasets, namely PASCAL VOC[15], and logo datasets possible to detected on line.

After analysis and introduction from that SDD model, a new algorithm MSSD model verification (modified single shot multi-box detector) aimed to increase both the accuracy and reliability of detecting small objects disappeared. Include in information about low-level network facilities for advanced rationality information, and multi-level features maps of predictable regression location and classification task inputs have been added to improve the need to explore reinforce. The VGG16 and deep residual network used in a new algorithm, and the feature maps of candidate box regression and classification task input were optimized.

• In detailed, we instantiate training our model with two backbone VGG16 and ResNet with two modules, element-sum and concat-sum modules.

This study will enhance the accuracy of target recognition by solving the problems of SSD method in target detection. It has important practicability in artificial intelligence, driverless, intelligent transportation, face recognition and other fields.

1.5 Structural Design of the thesis

This thesis report is structured into five chapters. Beginning with two (chapter 1, chapter 2) theoretical chapters. Since the convolutional object detection is a collection and combination of several fields in machine learning, discussing several related theoretical topics which seem desperate at first definitely is needed. proceeding from chapter 2, we will start with a short introduction to convolutional neural networks as a combination of computer vision and machine learning. Next, we introduce a short evolution of image recognition, we end the chapter by discuss how convolutional object detection used for the problem of detecting objects and review the popular recognizing an object methods. In chapter 3, we discuss detailed explanation with SSD method and our adaptations. In chapter 4, we will take a look at the datasets we’ve been working with, which are VOC2007 and Logo dataset, for image classification, also we shift to the experimental part beside discussing the details of the datasets and how will evaluated and shown the results we got during the implementation steps.

Ultimately, we provided the conclusion of our project in Chapter 5 and addressed, discussed in the future the recommendations and improvements on the project

CHAPTER 2

BACKGROUND AND RELATED WORKS

Since 1980, Artificial Neural Network (NN) has become a popular direction in the field of artificial intelligence research. First, the method abstracts the neural network of living objects from the direction of information analysis and processing, and then builds some simple models. Finally, these models are combined according to different connected methods to form different network structures. WSMcCulloch and W. Pitts established the MP model in the early 1940s [18]. Based on the MP model, a method for describing the mathematical concepts of neurons and describing the structure of neural networks was proposed, verifying that a single neuron also Being able to perform logical functions has opened up a new era of artificial neural network research, and artificial neural networks have experienced a long history of development since then. Until the early 1960s, the concept structure of the ”perceptron” was established by F. Rosenblatt. The structure is a neural network composed of multiple layers. Its design and production put the theoretical research of artificial neural networks into practical engineering. Subsequent years of investigation and development, the theoretical structure of artificial neural networks has achieved excellent results in a wide range of research areas. At the same time, a large number of neural network structural models are proposed, such as perceptrons, feedforward neural networks, BP neural networks [19], Boltzmann machines [20], convolutional neural networks, and so on.

2.1 Convolutional Neural Network (CNN)

Today, convolutional neural networks are presently among of the most outstanding algorithms of deep learning techniques utilizing the images datasets. Whilst relevant feature has to be manually extracted for traditional machine learning, which in deep learning certain features are learned by using raw images as input. A CNNs working on a layer of sends and receives also several hidden layers, convolutional, pooling and entirely linked films are examples like invisible layers. The CNN architecture differs in type and numeral of layers which had been implemented and introduced for its particular application. Classified responds, the procedure shall contain a group of operation layer, whereas the network should include a regression layer at the end of the network for continuous responses. Purposefully, each layer of CNN neurons is shaped and organized in a 3D structure, aiming to obtain a three- dimensional output through passing three-dimensional input throughout the CNN layer. Fig- ure 2.1 shows that the images are held by the input layer as 3D (width, height and depth- RGB as dimensions) inputs. Below, regions of the picture bind together to nerve cell within winding layer plus converted into such a three-dimensional output, look at figure 2.1.

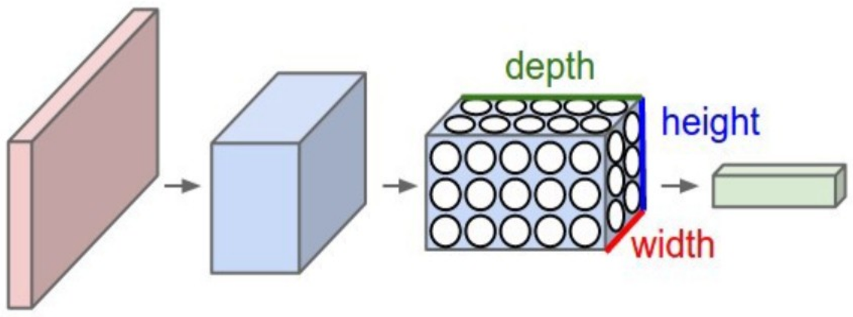


Figure2.1: A CNN which the red input layer consists of an image that converted into a 3D structure

Source: http://cs231n.github.io/convolutional-networks

CNN structure involves of many hidden layers. In each layer, nodes are involved while activation volumes are changed using of different functions. To create CNN configurations, there are four principle types of layers that are used, the example is shown in figure 2.2.

- Convolutional-Layer (CONV).
- Rectified-Linear-Unit-Layer (ReLU).
- Pooling-Layer(POOL.
- Fully-ConnectedLayer(FC).

2.1.1 Convolutional Layers

In CNN, convolution has been often familiar with output a characteristic design upon data, which can be the original image or other feature map. The main purpose of using convolution is to take advantage of the special structure of the input and learn how to transform it to the most informative form. In practice, convolutional layer's conduct is monitored in a series of hyper variables, which brings flexibility to the design of neural networks and allows to adapt themtovariousproblems:

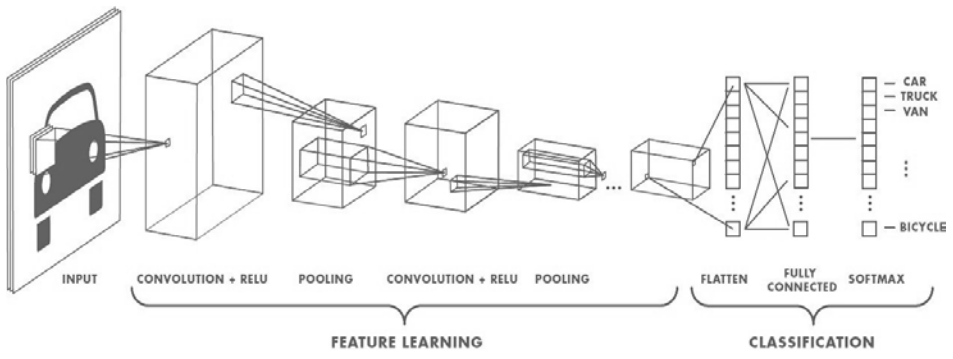


Figure 2.2: A CNN architecture uses to classifying an image as belonging to one of the categories

Source:http://cs231n.github.io/convolutional-networks

- Kernel size defines the attributes of complication core. Controls the input area where neurons are sensitive. It chose the right value for this parameter almost always according to a set of data. One way is to capture important detail by determining the smart size of the first row according to the scale of the images. Edges however, there is no rings for deep players theory is not end the maximum size of intellect is experimentally determined. In the case when the input contains multichannel images or any three-dimensional data kernel itself is often three-dimensional.
- Core number output dimension maintainers controls the number, because each core will produce a different main map. Increasing number of particles in the architecture can be frozen in the reduction of information less where the map size decreases with each layer. It also controls the capacity of the model under growing amount with kernels overall figure of trainable parameters grows up.
- Padding: convolution is unclear bordering inputs, because several segment of the grains does disproportioned few intake value it can consume. Addresses issue like angle cases to apply convention, the input possibly prepared under zero input value. Figure for which is determination directly influenced by the volume result, so wadding could used to control it.
- Sliding convention control pass the ascending price of two shall tell that after confusing several point, are skipped during dimension tour by manipulating, we overlap and decrease the output size of various accepted fields you can manage. Let

n_{in}, n_{out}, k, p, s be the total number of the inputs and outputs, total kernel size, padding size and the stride, resp. Then the following relationship holds true [16]:

$$\boldsymbol{n_{out} = n_{in} + 2_p - {}^{k}/_{2} + 1} \tag{2-1}$$

2.1.2 Pooling Layers and Dropout Layer

pooling operations always follow each convolution operation in order to control over fitting, minimize training time, and to further simplify the information. Receiving and compresses the input from previous feature in order to extract and condensed output feature map which feed forward to the next process is the core of the function of the pooling process. There are, as you know, two common pooling technique, maximum and average.

- Average pooling means calculating average between the field pooling values (every period from quality design).
- max pooling consider majority widely utilized form of pooling which select the greatest value between each patch of the feature map.

As shown, Figure 2.3, in a max-pooling operation, a 2 *x* 2 max-pooling filter induces future map reduction from 4 *x* 4 to 2 *x* 2.

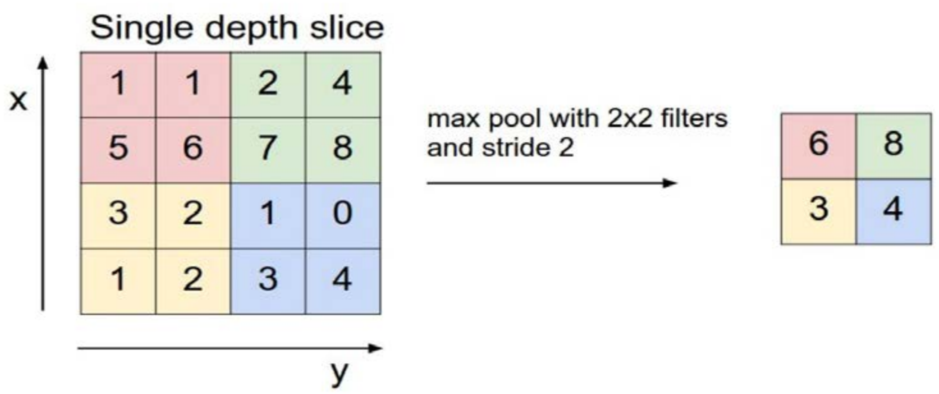


Figure2.3: Max pooling takes from each window the largest value

Source: http://cs231n.github.io/convolutional-networks

Dropout [37-39] was proposed by Hinton in 2012. Complex convolutional neural networks are very prone to over-fitting immediately instruct with tiny number set of information. For the purpose of to prevent with over-fitting phenomenon, people can enhance the comprehensive capacity of the series of neural networks by discarding some feature data, as shown in Figure 2.4. Assuming such an intake and result are x and y, accordingly, x is forward-propagated through the neural network, followed by the inverse reproduction formula is used to update parameters so that the error value approaches y. After adding the Dropout policy, the calculation process is as follows:

- Temporarily and randomly discarding one-half of the hidden neural units in the network layer, and the number of input and output neural units remains unchanged.
- Input x The forward propagation calculation is performed in the neural network with the Dropout policy added, and then the previously calculated error value is back-propagated in the modified neural network. After a small batch of training data samples have been performed, the weight w and offset b are updated using a stochastic gradient descent method in the neural unit that has not been discarded.
- Continue to repeat this process: recover the discarded neural unit, randomly select one-half of the hidden layer neural unit from the temporary and delete it, for a small batch of training data samples, use before Calculate the propagation, then use back- propagation to calculate the error loss value, and finally update the weight w and the offset b by the stochastic gradient descent method.

$$y_i^{l+1}=f(w_i^{l+1}(r^l * y^l)+b_i^{l+1}) \quad (2\text{-}2)$$

The role of the Bernoulli function is to generate a probability r vector, which is a vector that randomly generates a ”0” and a ”1”.

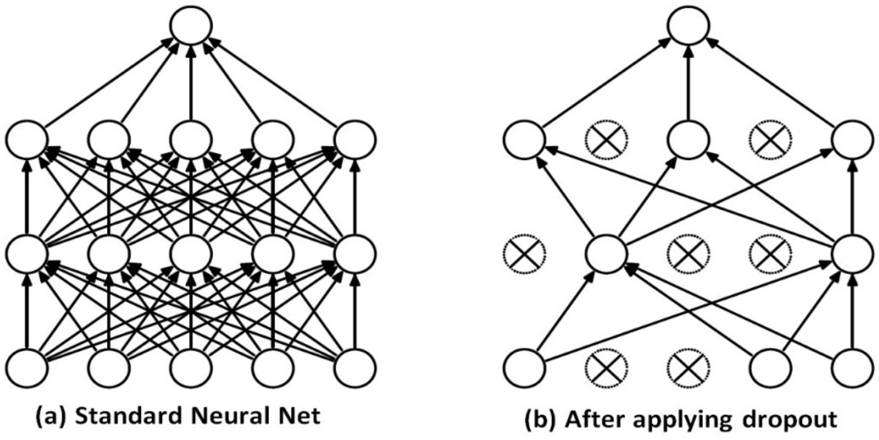


Figure 2.4: Dropout Neural Net Model

Source: http://laid.delanover.com/dropout-explained-and-implementation-in-tensorflow/

Summarize the advantages of the Dropout layer. First, for all the neural units of a hidden layer, set a probability value r during the training process, and delete some of the neurons temporarily and randomly. During the test, the weights are used (1-r) the probability value ensures that the same expected value can be used for each weight in the training and testing process. The other is that this layer strategy reduces the dependence of neural units on each other and makes some contribution to prevent over-fitting.

2.1.3 Fully Connected Layers

Completely linked sheets are a primary component of Convolutional Neural Networks (CNNs), which recently have proven noticeable success in recognizing and classifying objects in com- puter vision. The Fully Connected structure in a neural network is a set of layers which each one take the output of the prior layer (The output of CNN process)as input, turns them into a vector through ”flattens” them, each representing the probability that a particular feature belongs to a label which will be an input for the following stage. The objective and role of adding fully connected layer comprising sort the photo in one label by utilizing the output of convolution/pooling process. In order to determine and boost the most accurate weights that belong to a label, the fully connected layers must go throughout a process named ”back propagation”. Each neuron in (FC) layers receives(forward - backward) weights and gives priority to the most appropriate label. Finally, the neurons mathematically ”vote” on labels, and the classification decision is voting winner. An example is showing below the structure (layers and type of layers) needed to process an image. The more images are complex the more convolutional/pooling layers would be required.

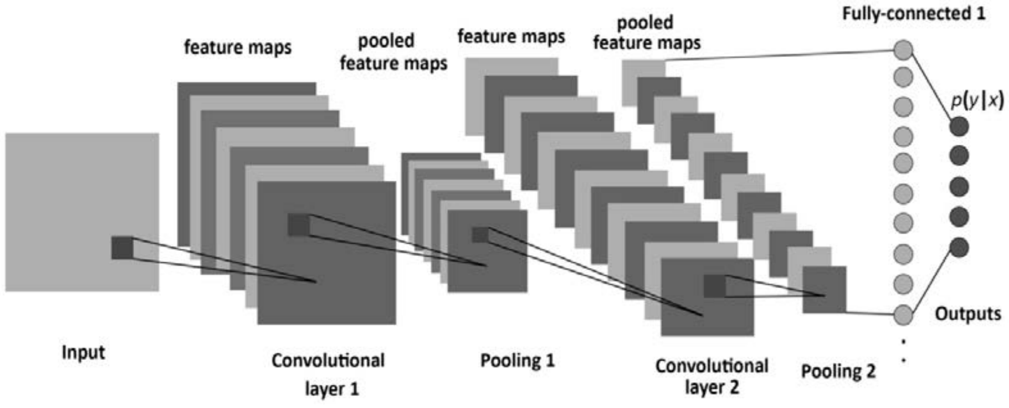


Figure 2.5: layers in a convolution network

Source: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

2.1.4 Common CNN Activation Function

The activation function is an important and indispensable part of the convolutional neural network. It can use linear and non-linear functions to enhance the expressive power of convolutional neural networks. Here are some common types of activation functions.

Sigmoid function: the Sigmoid [42] function is a non-linear activation function that is used more in the binary classification of things, and its definition expression:

$$\sigma(x) = 1/1+e^{-x} \qquad (2\text{-}3)$$

And its derivative :

$$\sigma(x) = \begin{cases} x;\ x > 0 \\ \sigma x;\ x < 0 \end{cases} \qquad (2\text{-}4)$$

The function graph is shown in Figure 2.6. As you can see from the figure above, the Sigmoid activation function is monotonically continuously increasing, and the value range is (0,1). Where x = 0, the function curve intersects the y-axis at 0.5. Analysis of the graph shows that the function easily reaches soft saturation, that is, the partial derivative at the limit value is zero, so it will cause the network model to easily cause the gradient to disappear when training the data.

- Tanh function: the Tanh function expression is as follows:

$$\sigma(x) = 2\,\mathrm{sigmoid}(2x) - 1 \tag{2-5}$$

Its derivative expression is :

$$\sigma^{1}(x) = 1 - (\sigma(x))^{2} \tag{2-6}$$

The graphic image of this function is shown in Figure2.6. It can be known that the tanh activation function is monotonically continuously increasing, and the value range is (-1,1). Where x = 0, the function curve intersects the y-axis at 0. Analysis of the graph shows that the function easily reaches soft saturation, that is, the partial derivative at the limit is zero. Although this function can converge faster than the Sigmoid activation function, it does not solve the problems existing in the Sigmoid activation function, which will also cause the gradient disappearance phenomenon when the network model is training data.

ReLU function: ReLU function [43] is the most widely used activation function at the moment and its function definition is:

$$\sigma(x) = \max(0, x) \tag{2-7}$$

The function image is shown in Figure 2.6. The ReLU function converges faster than the tanh function when training data, and provides sparse expression capabilities for convolutional neural networks. In terms of operation speed, the calculation of the ReLU function is simpler and cheaper than the exponential calculation of the logistic curve and the hyperbolic tangent function. Identify ReLU function is negative, its derivatives are all 1, when x is positive, its derivatives are zero, which can play a one-way suppression function, thereby effectively mitigating the disappearance of the gradient. Although the ReLU function has the above advantages, after all, it is not a big deal, and it also has some obvious disadvantages. For example, when the neural network model is trained when x is negative, the possibility of neurons disappearing is very high, so the dead neurons cannot update and calculate the weights and biases.

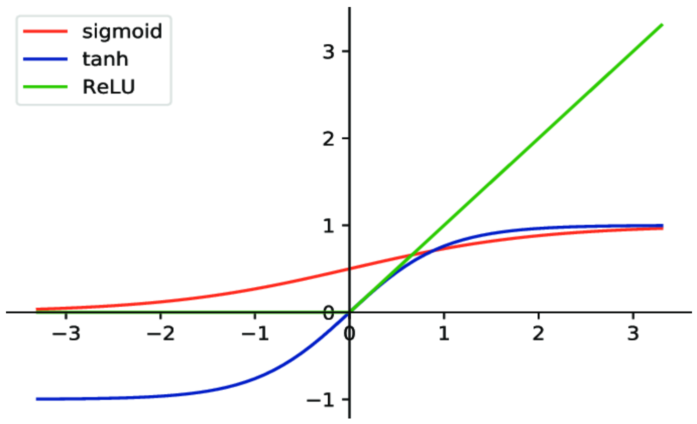


Figure 2.6: Function curves of sigmoid, Tanh and ReLU

2.1.5 Batch Normalization Layers

To make training much easily in a very deep network we use Batch-normalization [17] to normalizes the output (activation of last Convolution or Fully Connected layers). What batch-norm is about? It is a learnable or adaptive pre-processing that applying norm process on the output to normalize the mean and variance in order to achieve a list of advantages like regularization, permit higher learning rates, minimize dependency on initialization and improves gradient flow. Let X ⊆ D be a batch of inputs, then Algorithm 2.1 Compute the output y of the batch normalization layer.

$$\mu \leftarrow \frac{1}{x} \sum x \in x^{x}$$

$$\sigma^2 \leftarrow \frac{1}{x} \sum_{x\epsilon} x \, (x-\mu)^2$$

$$\hat{x} \leftarrow \frac{xi-\mu}{\sqrt{\sigma^2+f}}$$

$$y = \gamma\hat{x} + \beta \qquad \text{Scale and bais}$$

By centering and scaling the feature maps batch normalization makes the gradient com- putation more robust because the main goal of the layer is to discard the change in the distri- bution of hidden layers output, which can happen in the process of training. This simplifies learning and provides faster convergence toward the minimum. The values of $\gamma \in$ R and $\beta \in$ R are determined in the process of learning.

2.1.6 Deconvolution

Deconvolution has introduced recently by [”ZKTF10”], which basically orders to increase dimensionality from one input data. Perform , the process of convolution is basically transposed. Originally, deconvolution produces patches of $\rho_{(i,j)}^{(k-1)} \in R^{1*1*m(k-1)}$

accordance input, and applicable a mass matrix of $w^{k} \varepsilon R^{(k-1)} * K * K * m^{(k-1)}$ simply put it creates $k * k * m^{k}$ resulting every $1 * 1 * m^{(k-1)}$ patch and extends dimension (height and weight) of the input

2.1.7 Development

Among the first effective and successful deep neural networks was convolutional neural net- works. Built up in the 1980s by Fukushima, the Neocognitron biologically inspired[29] have provided A new model of translation-invariant object recognition in the neural network. Le Cun et al. merged a learning algorithm with that kind of method, i.e. back-propagation[33]. Almost all of these early approaches were used for handwritten recognition of character. The neural network methods disappeared prominently after attempting to deliver any promising results and entirely were substituted by support-vector-machines[47]. Then, Krizhevskky et al.[22] in 2012, accomplished fantastic outcome on the dataset of the ImageNet-Large- Scale-Visual-Recognition Challenge (ILSVRC) by incorporating Le Cun’s approach with previous approaches of deep learning fine tuning. These findings have popularized CNN networks, made it successful and led to discovery and development of new powerful meth- ods described later to detect objects. Simonyan and Zisserman[25] discussed and addressed the impact on position and clas- sification accuracy for the 2014 ImageNet challenge of expanding the depth of a CNN. Through the use of 16 and 19-layer deep convolution networks The team has accomplished outcomes that have made the state-of - the-art highly enhanced. The architecture of 16 layers involves 13 convolution layers, 5 max pooling layers and 3 FC layers, using rectified (ReLu) activations in All hidden layers. FC layers minimize 4096 channels to 1000 softmax outputs that regularized by dropout technique. The latest (2016) winner[2] in the ImageNet compe- tition of the category of object detection Is also CNN-based. The approach uses a mixture of production of CRAFT region proposal [45], clustering, Bi-directional gated CNN[46], ensembling and landmark generation.

2.2 Advanced Convolutional Object Detection

here, discussing and comparing different methods of object detection that utilize convo- lutional neural networks will be done. In particular, we will consider those methods that combine convolutional neural networks(CNN) with regional proposal.

2.2.1 R-CNN

In 2012, Krizhevskky et al.[22] had achieved favorable and promised results utilize CNNs the task of general image classification (look at section 2.1.5.). In 2013, a new method[6] called R-CNN ("CNN with area proposals") had just been published by Girshicck et al. which these results are generalized to object detection. R-CNN is a basic algorithm would propose a number of regions to bounding box objects detected using a selective[44] search algorithm and classify them one at the time. For each label it will output the label's name and the bounding box. As shown in figure 2.7, R-CNN algorithm has several stages. The first stage is extracting regions proposals which we use a region extraction algorithm (selective search[44] or other region generation algorithm) to propose or extract those regions. Next, the CNN features are extracted by convolutional network from each region proposed inde- pendently for classification, following by warping the " proposed bounding box" to a size fitted for CNN and then fed into the network. After the features has extracted, classifying what is the object is in this region through inputting the features to support vector machines (SVM) giving the final classification. The R-CNN is trained in several stages starting with the convolutional network(CNN) [8] and ending with classification algorithm SVMs to train the generating area proposition. Although R-CNN is considered very important process, because that is first practical solution to detect objects using CNNs. But it still suffering from many drawbacks Like expensive training, training consists of multiple stages and slow and etc. Slowness is one of the worst its drawbacks being needs to be processed by repeat

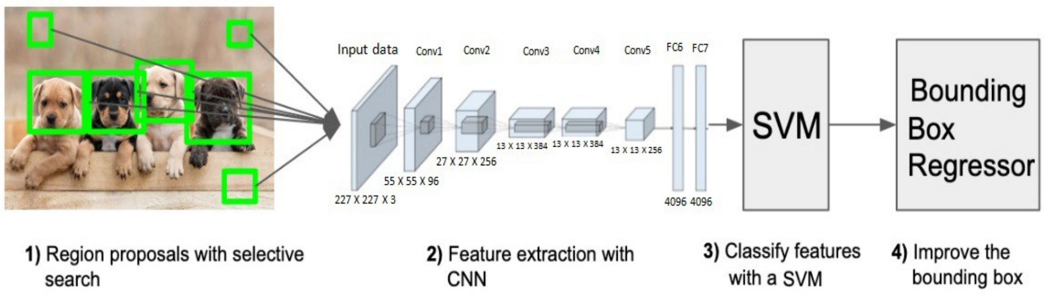


Figure 2.7: The R-CNN (Region with CNN feature) system

Source: https://towardsdatascience.com/r-cnn-3a9beddfd55a

the convolutional network 2000 times to extract feature for each image. And this is what are later methods have been improved .

2.2.2 Fast R-CNN

In 2015,Fast R-CNN [8] published by Girshicck, what this approach do is obtaining CNN features from each area proposal over the entire image and collating them within one CNN features matrix, The feature matrix is then forwarding and branched out to be used in next classification and bounding box regression phase. Figure 2.8 is illustrating the generic struc- ture of Fast R-CNN. An image with its computed regions of interest (ROI) is received as input in Fast R-CNN method. As the same with R-CNN, external algorithms are used to generate RoI like selective search [44]. The CNNs which include several pooling ,convolu- tional layers is used to process images.

The feature map generated from CNN after several convolutions and pooling layers is entered to a RoI layer. The output of ROI is a fixed-length feature vector for each region proposal from the feature map. The vectors are then feed forward to the FC layers which are connected to output layers:

- A softmax classifier of (k+1) classes producing discrete probability distribution per ROI or probability estimates for the object classes.
- A bounding-box regression which estimates offsets for K classes comparatively to the ROI.

Comparable to R-CNN, Fast R-CNN is faster being needs much shorter classification time per image up to a second on a state-of-the-art GPU [8]. This is primarily because the con- volution operation is done per image only once rather than fed 2000 region proposals to

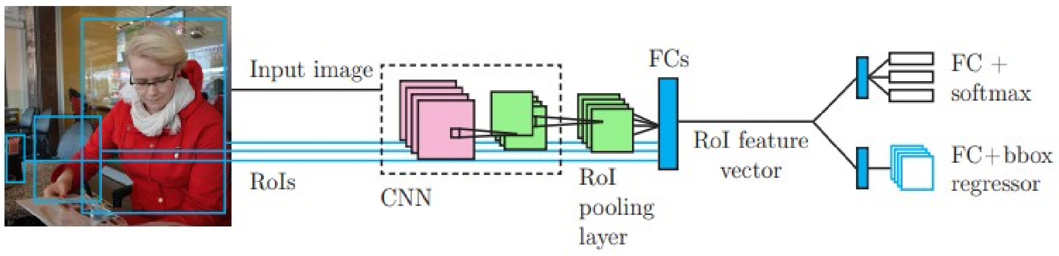


Figure 2.8: The Fast R-CNN forward computation

Source: **https://towardsdatascience.com/r-cnn-3a9beddfd55a**

CNN once at the time per image. As the decrease of detection time, the total calculation time begins to rely significantly performance of the method of generating the region pro- posal. So generating ROI form is the cornerstone or bottleneck[8] of the computational. In addition, when there are many RoIs , the time spent on assessing fully connected layers can takes control the time of convolutional layers. Accelerating the Classification time can be done by about 30% by compress the fully-connected layers utilizing the truncated singular value decomposition [8],This leads to a slight decrease in accuracy, however. According to [8], during training Fast R-CNN approach is much more effective than regular R-CNN up to nine-fold drop in training time. The whole network including (RoI pooling and FC layers) can be trained with the back-propagation and stochastic gradient descent algorithm. Typically, as a starting point we use a pre-trained network to facilitate training and then fine-tuned. Mini-batches approach of m images is used to train the network.

2.2.3 Faster R-CNN

The fundamental idea of Ren et al.'s, Faster R-CNN [21] is to utilize the shared convolutional layers for detection and for generating region proposal. The researchers found that feature maps that produced by backbone networks also could be used to collect the region proposals. Region proposal (RPN) is a fully-convolutional component of the Faster R-CNN network which produces the feature proposals. The authors announce the Fast R-CNN architecture as a successful end-to-end convolution network for the detection functions. Faster R-CNN network has already been trained by switching between generating and detecting RoI training. First, there are two different networks being trained. Such networks are then integrated and fine-tuned. During fine-tuning, some layers have been trained and some layers have been kept fixed in turn. A single image received by the trained network as input while the fully convolutional shared layers produce the feature map from the input image. While the feature maps feed to the RPN, regional proposals will be output of RPN that are input to the final detection layers include a (RoI pooling layer) with the said feature maps to- gether outputting the final classifications. Regional proposals have become almost cost-free computationally using shared convolutional layers. The advantage of being realizable on a GPU was added by generating region proposals by RPN. Traditional methods for generating RoI, including Selective Search, are implemented using CPU. The method uses special anchor boxes rather than a pyramid of scaled photos or a pyramid of varying filter sizes to handle different sizes and shapes of the detection window. Because of RPN faster R-CNN can propose regions with different size and the anchor boxes used as reference points to several region proposals which centered by the same pixel.

2.2.4 Single Shot Multi-Box Detector(SSD)

The Single-Shot-Multi-Box Detector(SSD)[10] goes even further with integrated detection. There is no need for the method to produce proposals, nor does it require any re-sampling of image segments. It uses a single pass of a convolutionary network to make object detection. The method begins with a default collection of bounding boxes, which very similar to a sliding window system. These absolutely include different scales and aspect ratios. The object predictions computed for these boxes involve offset parameters that predict how different from the default box are correct bounding box around the object. By using feature maps from various different layers of convolution (i.e. smaller and larger feature maps) as input to the classifier stage, the algorithm works with various scales. While the method produces a dense collection of bounding boxes, non-maximum suppression algorithm is following the classifier, which excludes most boxes under a certain confidence threshold. Although SSD method is less suited for small objects, it is one of the most efficient method and therefore being used as a method for object detection.

2.2.5 Non-maxima suppression

Non-maximum suppression (NMS) has already been used in several aspects of Computer vision. Its necessity comes from the imperfect of detection algorithms to localize the region of interest, resulting in groups of many detection near to the real location. So it’s a way to make sure that your algorithm detects each object only once. To implement (non-maxima suppression) follow the following :

- Discard all boxes with $p_c \leq 0.6$.
- While (loop):

o Pick the box with large p_c,output this as a production.

o Calculate the IOU between remaining box and output prediction p_c in the previous step and if IOU= $\{(iou \geq 0.5 \quad other)$

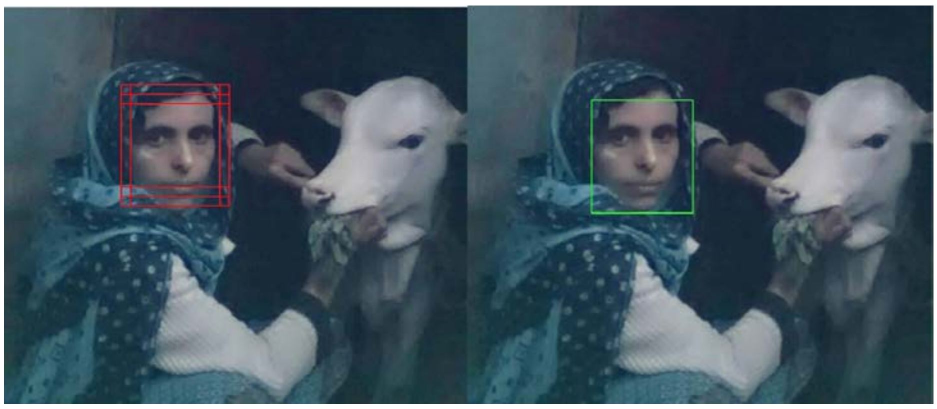


Figure 2.9: Process of Non-maxima suppression

2.2.6 Region Proposal Network

Taking the drawbacks of using selective search to extract regions of interesting in account, regions suggestion structure (RPN) created and used rapidly R-CNN method for minimize and mitigate problem of computational requirements of the overall inference process through determine where to look. The RPN efficiently and quickly takes all the anchors from every location and assess whether extra processing should be performed in a particular region by outputting K good bounding box proposals, each anchor has two different output(2 scores) represent the probability of existing an object or not at each location. The First one (classification) is the probability of whether or not the predicted box contains an object (background). An "object-ness score", if you will RPN uses the object-ness score to filter predictions in order to deliver good predictions for the next stage. Notice that the RPN ignore labeling object to which class, it only care about anchors which contain something looks like an object and not background. The second output is the bounding box regression to determine a predicted bounding box $P_{x_{\text{center}}}, P_{y_{\text{center}}}, P_{\text{width}}$, P_{height}. Far away from selective search RPN completely implemented in a fully convolutionar manner efficiently, utilizing from the base network output(feature map) as input. First, In Faster R-CNN they used a 3x3 kernel size with 512 channels that slides over a high-level conv feature map. We use a 1x1 kernel with k channels (k depends on the number of anchors) to get two parallel convolutional layers.

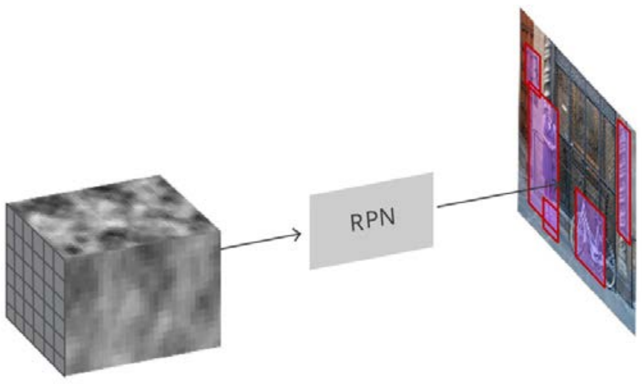


Figure 2.10: Convolutional implementation of an RPN architecture

Source: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

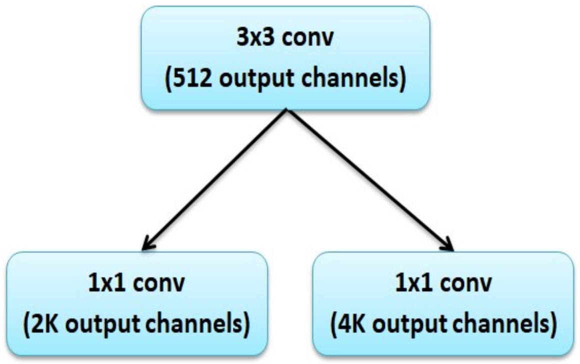


Figure 2.11: The RPN generates proposals over the image

Source: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

The confidence scores in classification layer(cls), each anchor has two predictions as output : the score of containing an actual object or not(background). For the regression layer (reg) that computes the box offsets, there are 4 predictions as output:$\Delta_{xcenter}$, $\Delta_{ycenter}$, $\Delta width$, $\Delta height$

That gonna be implemented particular offer presenter in order to get the final proposals. The final good set of proposals that will be further processed are those anchors with a corresponding high ”objectness” score.

2.2.7 Compare the Methods

Above in section (2.2), we have mentioned the qualitative leap of speed and accuracy between regular R-CNN and Fast R-CNN. How about their performance by Fast R-CNN com- pared with advanced methods (SSD, Faster R-CNN)? Comparing the performance of advanced methods(Faster R-CNN, SSD) and Fast R-CNN on standard dataset PASCAL-VOC-2007 test set has done by Liu et al[10]. On PASCAL-VOC-2007 training data, Faster R- CNN performance slightly improving than Fast R-CNN which achieves 69.9, to 66.9, mean average precision respectively. Also SSD had achieved 68.0 mAP with 300 x 300 input size and 71.6 mAP with 512 x 512 input size. Faster R-CNN and Fast R-CNN standard implementations are resizing the length of the input picture some smaller size of 600 pixels. It seems that SSD achieves better performance with images of similar size. However, to achieve such result [36], SSD needs or requires an extensive utilize of data augmentation to expand the precision with accuracy of recognizing for things. Fast R-CNN and Faster RCNN utilize only horizontal flipping technique while SSD benefit from the variety of the data augmentation techniques. Currently it is unknown, whether Faster RCNN and Fast R-CNN would utilize and benefit from additional augmentation. Undoubtedly, advanced methods (SSD, Faster R-CNN) are considerably more precise and correct than Fast R-CNN but the surprise improvement is the speed. After eliminating most of the low probability detection through the non-maximum suppression, SSD512 frame rate drops to 22 FPS while SSD300 frame rate drops to 59 FPS. Meanwhile, Faster R-CNN based on a VGG-16 backbone can run at 7 FPS [10]. The running time of Faster R-CNN [21] has reported 5 FPS i.e. by the premier authors 0.2s per image. In case of evaluation speed, Fast R-CNN and Faster R-CNN both have approximately the same speed but Fast R-CNN requires extra time for generating region proposals. Generating regions using Selective Search in Fast R-CNN require 2 seconds per image as mentioned by the authors.

2.2.8 Speed Accuracy Trade-off

With the brief overview of the most widely used object detectors, selecting an object detection method to experiment with can be challenging. However, "Huang et al" has made a comprehensive study of all the above mentioned object detectors and compared them in multiple aspects such as, speed, accuracy and memory. This paper has been a foundation for this work in selecting the network architecture. The study concludes that SSD has one of the better trade-offs between speed and accuracy.

2.3 Open Source Framework For Object Detection

Open source framework for object detection, such as google Tensor Flow Object Detection API, OpenCV's DNN library and Microsoft Cognitive Toolkit (CNTK), we are going to develop object detectors related to our application grant. Open source frameworks provide a pre-trained object detection method that can easily correct our data sets.

2.3.1 TensorFlow Object Detection API

The Google Tensor Stream Object Detection API is an open source object detection framework. It is based on tensor flow and for the user to define, train and use the Permite object detection model. Developed by Google Brain Team, TensorFlow Open Source Software Library Uses data flow charts for digital computers.[1]. Two key components, nodes and edges in the data flow diagram are. The nodes represent the mathematical calculation (operations) and the edges represent tens of (multidimensional) that flow between nodes .Two key components, nodes and edges in the data flow diagram are. The nodes represent the mathematical calculation (operations) and the edges represent tens of (multidimensional) that flow between nodes How Parameters and Output Change during Tensor Board Comographic Viewing and Model Building Provides a web interface to understand. As mentioned earlier, the Tensorflow Object Detection API provides multiple pre-trained models such as Mobile Net's SSD model, ResNet's Faster R-CNN model, and ResNet's R-FCN model on different datasets. To fine-tune our individual object detectors, we can choose to initialize our training from among the pre-trained models. The choice of the pre-trained model depends on the intention of our application it gives API ideas for the speed and accuracy of various item tracking models. Acceleration is the most important requirement in real-time modern object identification applications Models trained with SSD and RFCN networks are pretty good, but low cost accuracy. Conversely, fast R-CNN is more reliable but more expensive over time from trained models. Figure 2.12 shows the balances for speed accuracy for different object detection models.

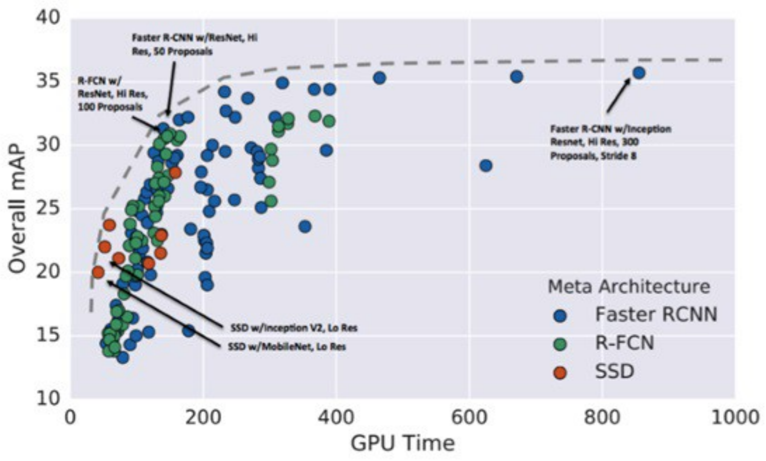


Figure 2.12: mAP vs. GPU time for different meta-architectures

Figure 2.12 shows that the most reliable of all is the Faster-RCNN model with Inception ResNet with 300 propositions but also the most expensive. The models equipped with SSD and R-FCN are however faster but less accurate. In addition, Faster R-CNN can be fast enough if the regions of interest with ResNet are small as is evident for Faster R-CNN with 50 proposals

2.4 Summary of This Chapter

This chapter analyzes and introduces in detail the origin of neural network and its development process. The paper focuses on the principle of convolutional neural networks in deep learning, and details the network layer, inter-layer relationships and parameters, built- in functions, and algorithms in the architecture. Then the open source frameworks for object detection and speed-precision trade-off for different object detection models have defined.

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

In this section we first give and present an overview for the used and proposed model of detecting small objects from an input image, introduce the SSD briefly which is a commonly uses object detection framework and then demonstrate in detail MSSD modules, which take advantage of the useful local context for late classification and regression layers. The loss function used for this task will be discussed separately. We will conclude this chapter by describing the image pyramid structure used by Hu and Ramanan (2017), how it fits in the SSD framework and why we expect it to be beneficial object detection.

3.1 Single Shot Multi-box Detector Network

The SSD network by Liu et al. is one of the architectures used most commonly for detecting objects. The network is Fully Convolutional and can therefore be used for images with any resolution. Two architectures are proposed in the original paper: 300x300, input resolution architecture and 512 x 512 pixel input resolution architecture. Our baseline model is SSD300 because its default model and original document is described by Liu et al. (2016). Furthermore, other papers also use this network as a baseline Zhang et al. (2017b). One property of the SSD network found by Huang et al. is that it's more efficient than other detectors and an efficient network was one of the constraints for the Sightcorp application. The SSD network is called Single-shot since both Localization and classification of objects was carried out through the network within a single feed-forward. This is contrast to, for example, the Faster-RCNN network Ren et al. (2015), from which it differs since it does not have a separate regional proposal network. Furthermore, the SSD network combines multiple feature maps with different sizes to generate predictions, similar to Hariharan et al. (2015), to be more scale invariant to objects. These combined predictions from the multiple feature maps produce two output, a bounding box offset and class confidence. The network consists out of three parts, a base network, SSD layers and prediction layers attached to multiple feature maps in the network.

3.1.1 Base Network

These first layers are called the base network and the base network consists of stacked convolutions to decreasing size. The purpose of this base network is to provide response maps that enable detection different sizes. The base network can be seen in figure 3.1 and is represented the convolutions conv1 till conv5. To be consistent with the original paper we use a truncated (fc6 and fc7 removed) VGG16 base network and initialize those layers with image-net weights. However, as mentioned by the authors one could replace the base net- work by any standard or non-standard architecture, e.g inception (Szagedy et al., 2015) or resnet (He et al., 2016).

3.1.2 SSD Layers

Subsequently to the base network, additional convolutional layers are added: conv6, conv7, conv8, conv9, conv10 and conv11, which are initialized with a truncated normal distribution. These additional convolutions are highlighted as SSD layers in figure 3.1. Similarly to the base network, the decreasing size of the feature helps with generating response maps for various object sizes. These layers however have bigger receptive fields and this helps to detect larger objects.

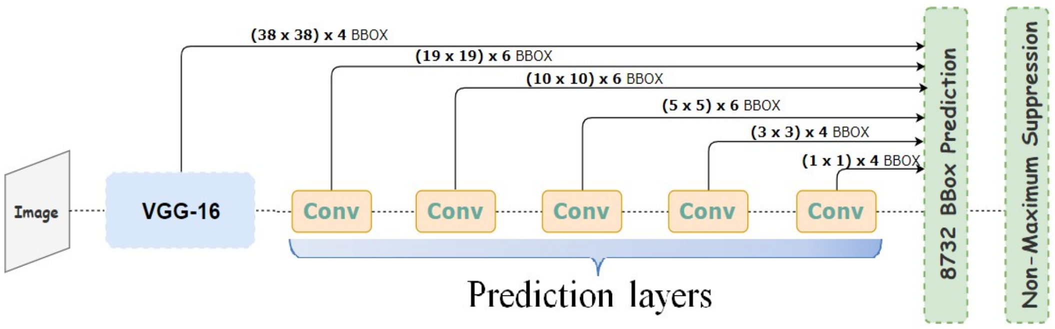


Figure3. 1: SSD network architecture

3.1.3 Prediction Layers

The prediction layers are attached to the convolutional base network and SSD layers (Figure 3.1). For the feature layer of size m x m x c, while m is volume of feature map and c is channels range. Convolutional layer is attached with a 3 x 3 x r x(classes + off-set coordinates) kernel, where r is the default bounding boxes number and the classes number is 2(classes and background). This kernel produces both a classes confidence, background confidence and also the offset of bounding box pertain to a default bounding box, which we will touch upon shortly. These prediction layers are attached to multiple points in the convolutional base network and SSD layers, namely conv4-3, conv7-2, conv8-2, conv9-2, conv10-2, conv11-2. More fine details are captured by lower layers and are able to capture smaller objects, while higher layers capture more semantically meaningful information and capture larger objects. Therefore, attaching multiple feature layers should help to capture the differently sized objects. All the prediction layers are concatenated at the network end, which will result a single output layer with a fixed number of predictions of the bounding box. The network parameters are shown in Table 3.1. As shown in Table 3.2, it is the

Table 3. 1: SSD network parameters.

Layers Name	Convolution	Input(w,h,c)	Output(w,h,c)
VGG Network Convolution Layer part(SSD is also prediction from Conv_4)			
Conv6(VGG FC6)	1024,3x3,1,6,dil=6	19x19x512	19x19x1024
Conv7(VGG FC7)	1024,1x1,1	19x19x1024	19x19x1024
Conv8_1	256,1x1,1	19x19x1024	19x19x256
Conv8_2	512,3x3,2,1	19x19x256	10x10x512
Conv9_1	128,1x1,1	10x10x512	10x10x128
Conv9_2	256,3x3,2,1	10x10x128	5x5x256
Conv10_1	128,1x1,1	5x5x256	5x5128\
Conv10_2	256,3x3,1	5x5x128	3x3x256
Conv11_1	128,1x1,1	3x3x256	3x3x128
Conv1_1	256,3x3,1	3x3x128	1x1x256

parameters of the loc network. The loc network is the network used to calculate the regression box. In the loc network, 4x4 and 4x6 are due to the prediction frame used by the loc network to predict the target. The prediction frame consists of (cx, cy, w, h), Representing the focal points and height of focal points and forecast frames. Multiply by 4, 6 because in this feature map, each feature point is divided into 4, 6 detection frames, respectively. Finally, the output dimension of the loc network is transformed into (Batch-size,-1, 4) to obtain the prediction frames of the 8732 targets in the detection frame.

Figure 3.3 shows the conf network parameters. The conf network is the network used to calculate confidence. In the conf network, 21 is a classified category, because the VOC dataset is classified into 20 categories. When doing a target detection task, a background category is added, so it is 21 categories. 4 and 6 are the number of detection frames divided by every feature spot of feature map. The output dimension concerning conf network is

Table 3. 2: SSD regression box prediction loc network

Serial Number	Convolution Operation	Input(w,h,c)	Output(w,h,c)
1	4x4,3x3,1,1	38x38x512	28x28x16
2	4x6,3x3,1,1	19x19x1024	19x19x24
3	4x6,3x3,1,1	10x10x512	10x10x24
4	4x6,3x3,1,1	5x5x256	5x5x24
5	4x4,3x3,1,1	3x3x256	3x3x16
6	4x4,4x4,1,1	1x1x256	1x1x16

(Batch-size, -1, 21) to obtain the classification score corresponding to the targets in the 8732 detection frames. Small objectives through SSD to improve identity, the

Table 3. 3: SSD confidence prediction Conf network

Serial Number	Convolution Operation	Input(w,h,c)	Output(w,h,c)
1	21x4,3x3,1,1	38x38x512	38x38x84
2	21x6,3x3,1,1	19x19x1024	19x19x126
3	21x6,3x3,1,1	10x10x512	10x10x126
4	21x6,3x3,1,1	5x5x256	5x5x126
5	21x4,3x3,1,1	3x3x256	3x3x84
6	21x4,3x3,1,1	1x1x256	1x1x84

parameters of the Pool5 layer of the VGG network were changed from 2x2,2 to 3x3,1. This can make the feature map after pool5 maintain a high resolution, Which is suitable for detecting small targets. Fully connected layers FC6 and FC7 are converted into convolutional layers, and the Atrous operation is added to Conv6. The Atrous operation is also known as hole convolution. This entity is presented from a large acquisition area without shaping the map.

3.1.4 Default Bounded Boxes

The selective-search algorithm by Uiijlings et al. has been a vital component in object detection methods in order to obtain region proposals. However, the SSD network has another method for this purpose. The

SSD network regresses a grid of default bounding boxes to fit the objects in the dataset. This grid of default bounding boxes is constructed as follows. For each feature map that has a prediction layer attached, the bounding boxes is tiled on each feature map cell, which means that every cell of the feature map will have a default bounding box that is centered in the feature cell. The center can be computed as follows,

$$x_i = \frac{i+0.5}{f_k}$$

$$y_i = \frac{i+0.5}{f_k}$$

where f_k is the length of the size of the feature map square and i and j range from 0 till f_k. The original model uses different ratios for their default bounding boxes, as could be seen in figure 3.2.

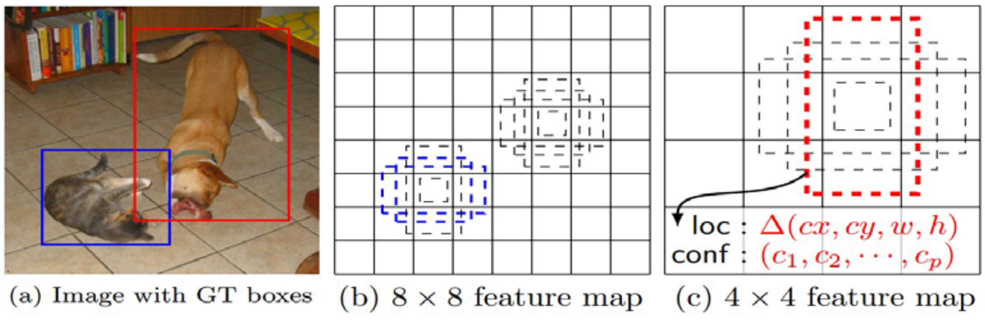


Figure3. 2: SSD network architecture

Source: https://towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca

3.1.5 Deep Residual Network (ResNet)

As the number of layers in the network increases, the problem of practice becomes more prominent. A more significant problem is the vanishing / explosion of the gradient, which affects convergence in the beginning. On the basis that as network awareness increases, the more depth network can change, the accuracy rate starts to decrease, which decreases the network the problem is called. Increasing the number of layers on a given network will increase the training error. Consider a shallow network architecture and a deep network built on it. Under extreme conditions, if all the layers added are a direct copy of the previous layer, this situation the training error of the lower deep network should be equal to the shallow network. Therefore, the root cause of network degradation is still the optimization problem. In order to solve the optimization problem, a residual network is proposed. The residual network is to add some shortcut connections to the forward network. These connections will skip some layers and pass the original data directly to the subsequent layers and will not increase the parameters and complexity of the model. In theory, the network is always in an optimal state, and the performance of the network will not decrease with increasing depth.

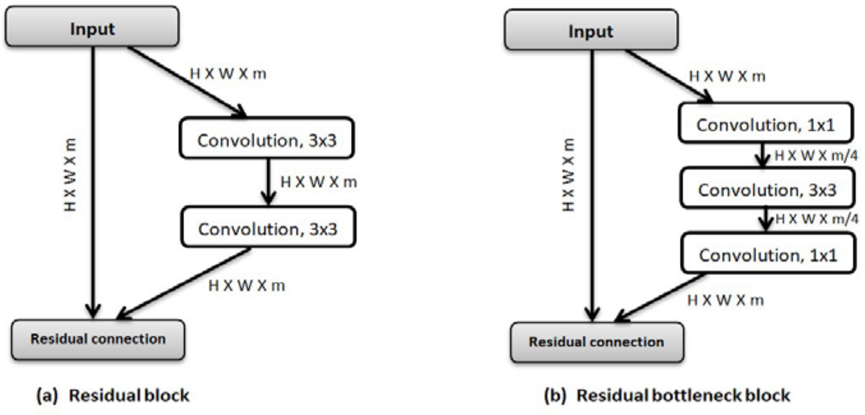


Figure3. 3: Residual locks, Viewed as suggested by [HZRS15].

For simplicity, BN, ReLU non-linearity, and identity mappings have hidden

[HZRS15] had introduced two forms of residual blocks and the whole structure is generally referred to basic components. The first is named a residual block (Figure 3.3, left), containing of two operations of convolution, a residual connection has been made between entry and exit blocks. The alternate call is residual bottleneck block (Figure 3.2, right), containing of three operations of convolution. Starting by a 1*1kernel to reduce the channels number and a 3*3 kernel, ending by applying the third 1*1 kernel to enhance the amount of volume. Residual blocks be required to instruct deep structure until 34 layers. For the deep networks which have over 50 layers, residual bottleneck block have resorted. In case of achievement these residually blocks on Image-Net dataset, 50-layers network achieves 22.85% top-1 error rate using residual bottleneck and 34-layers network achieves 25.3% top-1 error rate using residual blocks. For our part MSSD with ResNet model,

we opted using the residual blocks because of the Extremely expensive of using 1 �1 convolutions in residual bottleneck blocks and we cannot cut down their cost through the use of an substitute process. In addition, residual bottleneck blocks are practically not just a perfect proper less layered correlate reason they do not extend the receptive field as strongly as residual blocks.

3.1.6 Very Deep Convolutional Network VGG16

The VGGNet model structure [31] consists of a base layer, an activation function ReLU, a pooling layer, a full link layer, and a softmax function. It has a total of 5 convolutional segments. Each convolution consists of from 2 to 3 convolutional layers followed by a maximum pooling. The number of convolution kernels also increases with the number of layers. Moreover, VGGNet uses a data augmentation method to prevent overfitting of the model. Its biggest feature is to use a small size Filter instead of a large size Filter. Its structure is shown in Figure 3.4. The input image is $224 * 224 * 3$, which becomes $224 * 224$*64 through two convolutional layers, and $112 * 112 * 128$ through one pooling layer and two volume base layers. $56 * 56 * 256$ through one pooling layer and three convolutional layers, 28 * 28 * 512 through one pooling layer and three convolutional layers, and one pooling layer and three convolutional layers The base layer becomes 14 * 14 * 512, and it becomes 7 * 7 * 512 through one pooling layer. It becomes 1 * 1 * 4096 through two full-connected layers. It becomes 1 * through a full-connected layer and soft-max output layer 1 * 1000. In this thesis, VGG-16 is used as the base neural network(backbone) architecture, as described in [Parkhi et al., 2015]. As you can see from the Figure 3.4 The network has 13 convolution layers and 3 FC layers. VGG-16 has already been used as the backbone because of its strong, reliable performance in the tasks of classification, meanwhile for its popularity for issues where learning transfer helps highly to improve results. Figure 3.4 illustrate different layer types correspond to different colors, the name and output size for each layer in the network is listed. In VGG-16 convolutional layers share the same kernel width and height $m_f = n_f = 3$ the same stride $s_w = s_h = 1$ and the same zero padding $p_w = p_h = 1$ Even pooling layers do the same through share the same format, with 2 * 2 grid size, and a stride of $s_w = s_h = 2$ ReLU function used as an activation function in VGG-16 as shown in Equation 3.3 and softmax loss as the loss function.

$$f_a(x) = \max(0, x) \qquad (3\text{-}3)$$

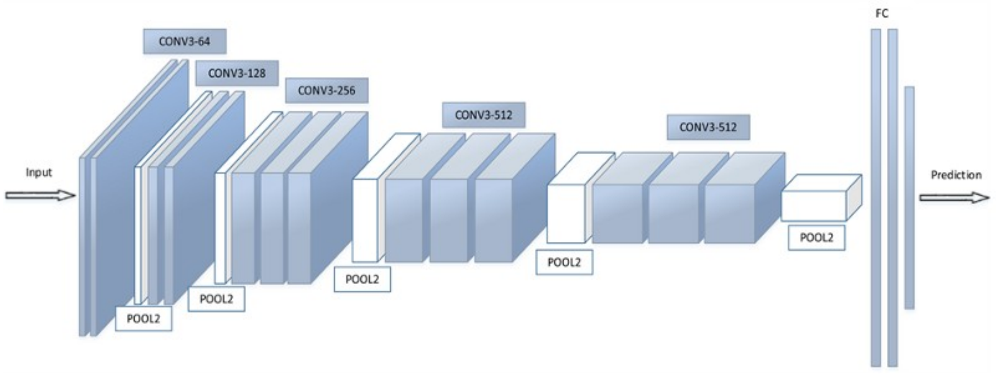


Figure3. 4: VGG-16 architecture for Classification and Detection

3.2 Modified Single Shot Multi-Box Detector (MSSD) Model

From the SSD model introduced and the knowledge of the deep residual network, the re- searcher found that with the deepening of the depth convolution network level, the detection of objects on the training set will appear to decrease in accuracy. So in order to correct about defects from SSD model, the new algorithm proposed is better than the traditional SSD model in terms of detection accuracy. In view of the two defects of the SSD model above, the methods are introducing the two improves modules (Concat-sum module , Element-Sum module) using the deep VGG16 and the deep residual network with the feature pyramid net- work module are adopted, so the MSSD model is more accurate than the traditional detection accuracy. In image classification functions, outstanding networks have proven to be better than VGGS because it provides escape links between criminal blocks, thus gradually putting an end to hits and reducing the way the network is moving deeper. In fact, ResNets typically can go up to 101 layers whereas VGG networks can only go up to 16. Because deep nets are often better for sorting images, ResNets are generally more accurate than VGG. Here we will apply our idea on both of the VGG and ResNet networks showing the result that we got in our experiment later. The shallower layers (conv-3) suffering from lacking of sematic information, so in order to compensate that lack we inject contextual information from other layer and come out with new design model named element-sum and concat-sum model . Although replacing the VGG-based feature extractor in SSD with ResNet-101 does not lead to greater performance. Hence, a custom-made prediction module is needed.

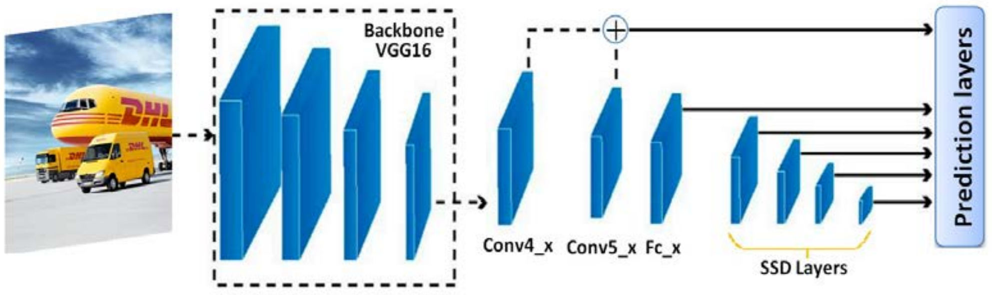


Figure3. 5: Feature-fused SSD architecture

3.2.1 Network Structure Based an Deep VGG16 Network

The author of SSD architecture choose VGG16 as a base network to obtain feature maps in order to feed them forward into next detection layers. In MSSD proposed model, instead of classify the normal ConvNet feature map, we exploit the pyramidal feature hierarchy in convolution layers before feeding to the detection layers. The proposed model is shown in Fig. 3.5. Deeper layers are used to predict larger objects, while shallower layers are used to predict smaller objects, thus reducing the entire model's predictive burden. Shallower layers, moreover, often lack receptive files which is an essential supplement for detecting small objects. Therefore, passing back the receptive files (semantic information) to the earlier layers captured in convolutionary forward computation will improve and enhance the performance of detection especially for small objects Which layers should be combined? We take advantage of the appropriate Conv-layers to provide helpful contextual information as extra-large receptive field often would absolutely introduce useless, large noise in the background. For large objects in deeper layers we don't use the feature fusion module in order to keep up speed as long as SSD uses their shallower layers, as in the case of conv4 3, to predict small objects. Suitable characteristics of different layers to select fusion layers specialized receiving areas use the mallet development method to search.

In a bid to inject the contextual information into shallower layers (such as conv4-3)

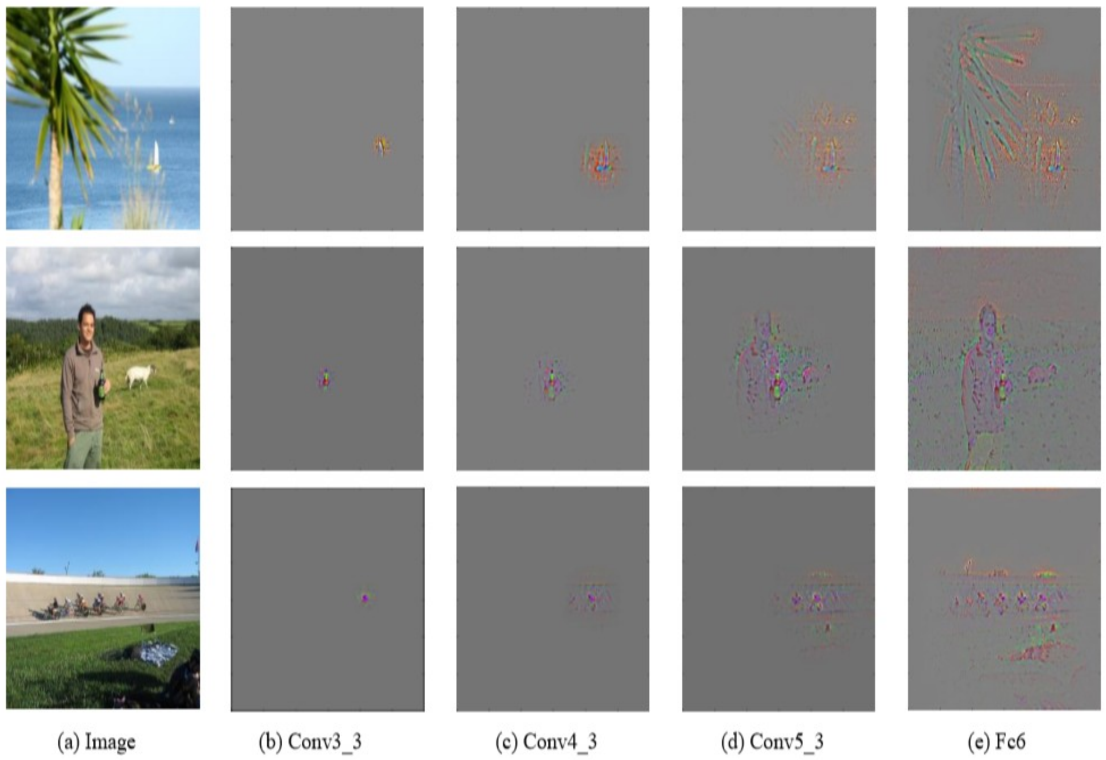


Figure3. 6: Layers show the useful receptive fields in SSD architecture

which lacks to the information about the semiconductor, MSSD model with two different feature fusion has designed named element-sum and concat-sum modules.

Concat-sum Module:

The MSSD with concat-sum module is shown in Fig. 3.7 A new technique named deconvolution is actually layers which used to render the feature maps of two layers, in our case conv5-3 and conv4-3, have the same size. As it's clear in fig 3.7, two 3x3 convolutional layers are used after conv-layers for learn to fuse the better features. Then layers of normalization follow with different scales before they are concatenated around their channel axis. The final fusion-feature-maps are generated as well as feature recombination by a 1*1 range convolution layer for dimension reduction.

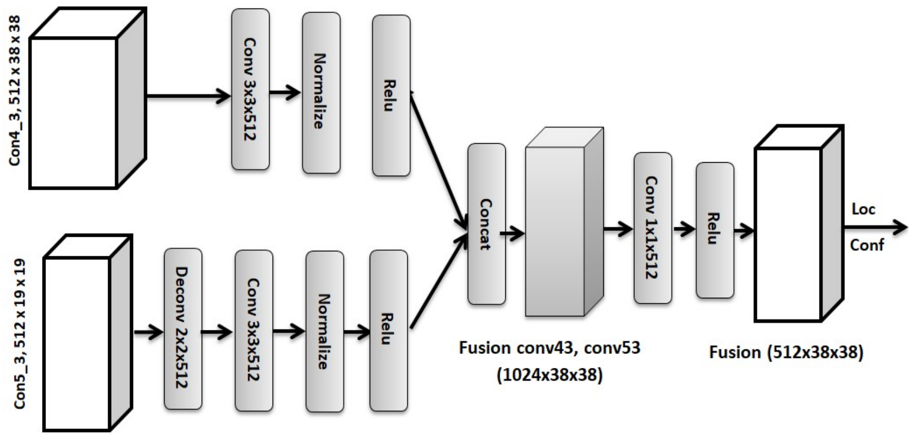


Figure3. 7: Illustration of the smart concat-sum model

Element-Sum Module Except for the form of fusion, it is the same as the concat- sum module. Two different feature maps layers containing different level features are summed up to equivalent weights point to point in this module. This process works perfectly in practice because of the two previously used convolution layers, which learn features adaptively from the two layers, conv4-3 and conv5-3, for better fuse results. This module takes inspiration from residual-101-based DSSD[32] which exploit the learned layer of de-convolution and elementary operation. concat-sum module

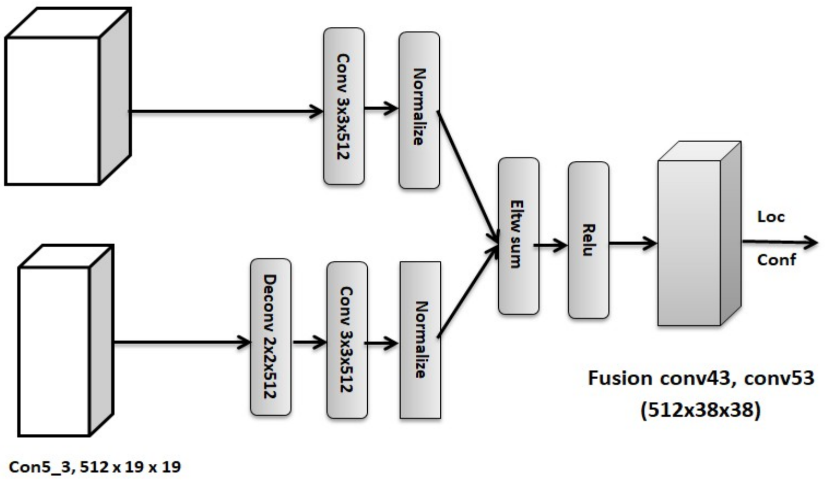


Figure3. 8: Illustration of the smart element module

incorporates multi-level functionality with the learned weights implemented by 1*1 convolution layer, while element-sum module uses manually set equivalent weights. With this distinction, element-sum module can improve contextual information's importance and the concatenation module can decrease the interference that caused by unnecessary background noises.

3.2.2 Network structure based on deep Residual Network

The literature shows that as the number of VGG network layers increases, the problem of training becomes apparent. The more significant problem is gradient vanishing/explosion, which affects convergence at the outset. Under the premise that the deep network can converge, with the increase of network depth, the correct rate begins to saturate or even decrease, which is called the degradation of the network. Increasing the number of layers on a given network increases the training error. The root cause of network degradation is still an optimization problem. To solve the optimization challenge, we use the residual network. Residual networks can be interpreted as making a kind of shortcut connections to the for- ward network layers. These connections skip certain layers and directly pass the raw data to the later layer. This research uses the ResNet101 network which is taking advantage of the residual network in the forward network to add some quick connections. The new shortcut connections don't increase the complexity and parameters of the model to improve the speed beside accuracy of detection objects.

Element-Sum Module same with concat-sum and element-Sum Module fusion module above we try to implement it with ResNet network with additional improvement on FPN (Fig 3.10) and predection layer. To achieve better accuracy, deconvolutional layers are used to increase the resolution of feature maps. The detection is then done using the "super-resolved" feature maps. In addition, to integrate information from earlier feature maps, deconvolutional modules are used. Technically, deconvolutional layers need not be used at all. Up-sampling followed by a convolutional layer can also achieve the desired effect. However, since up-sampling layers do not have any learnable parameters, it may not lead to the optimum results. Hence, de-convolutional layers are used.

- **PredictionModule**

The prediction model structure is shown in Figure 3.10 which is the method used by SSD, and the multi-scale prototyping of the system is extracted directly to make the prediction of classification and box regression. Figure 3.9 is the network structure of the Res-net residual unit, during the training phase.

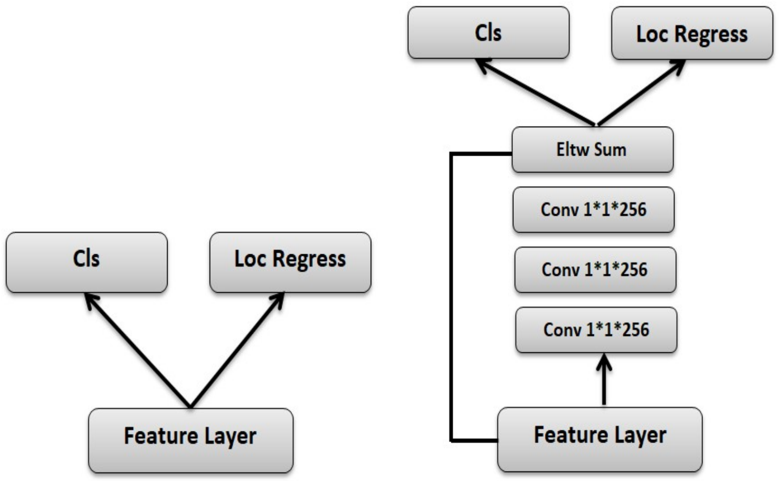


Figure3. 9: The framework of classification and regression of SSD and ResNet

For the forecasting phase, this research uses the (right-Figure3.9) approach to the work of the featuremap. This module is added to this research because multi-scale CNN indicates that improving the performance of each subtask can improve accuracy.

- **Feature Pyramid Network Module**

The feature pyramid network module refers to the fusion module of the upper and lower features in the MSSD, and the basic structure is as shown in Figure 3.10. This research builds the network in the form of FPN, followed by three BN (Batch Normalization) and three 3x3convolution, here convolution also acts as a buffer to prevent the gradient from affecting the backbone network too severely and to ensure the stability of the network. After feature fusion, each feature layer channel dimension responsible for prediction changes to 512. BN operations are placed between the convolution layer and the activation layer, and the top sampling of some of the previous methods is achieved by two-wire interpolation. MSSD is a top sample feature map learned by the feature pyramid network module. Low-level feature maps require increased regularization of operational processing because their feature maps are different in size and other layers and can be difficult to train in practice if they are mixed together. And it is important to note that the data size of different layers is also different, so it is not possible to directly merge. Therefore,L2 regularization is used. The lower-level feature needs to be positive when MSSD is used for high and low layer feature fusion.

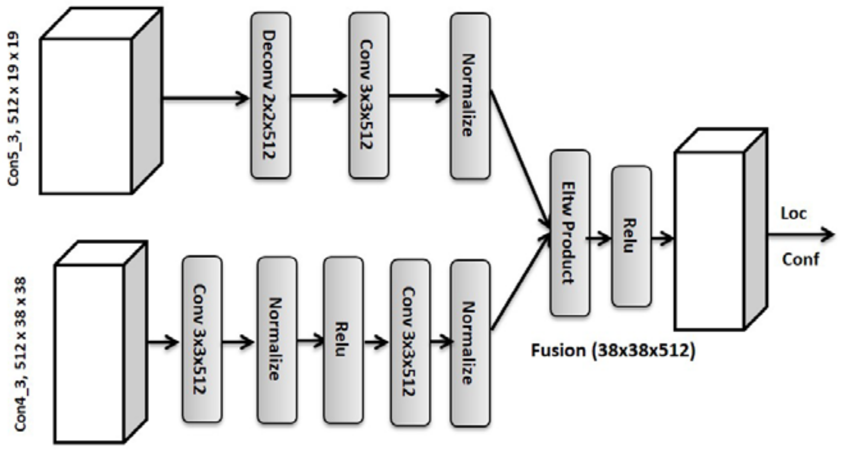


Figure3. 10: Smart Feature Pyramid Network Module.

To include more advanced context in the detection, transfer the prediction to the layers of the series of feature pyramid networks after the original SSD setting. Add an additional feature pyramid network layer to continuously increase the resolution of the feature layer.

3.2.3 Model training

- MSSD based on deep VGG16 training Method:

  We train the proposed MSSD fusion models, concat-sum and element-sum, on the both logo and PASCAL VOC2007/2012, that contain 20 class in 9,963 and 22,531 images, respectively. For further 10K iterations both of the two function fusion models are fine-tuned to the well-trained SSD baseline. The learning rate has chosen to be $1 * e^3$ for the first 60K iterations and then decreases to $1 * e^4$ at the 60K and $1 * e^5$ at the 70K iterations.

- **MSSD based on deep ResNet training Method**

In the framework of caffe[38], the basic network of SSD is changed to ResNetl01 and then a new SSD model is retrained. Collect data set of VOC2007[39] as an sample. The data used in the training set is the VOC2007 data set. The test set of 07 is used, and a total of 7k iterations are used during training, and the learning rate is $1 * e^3$. In the first 4k iterations, then adjust the learning rate to $1 * e^4$, $1 * e^5$ and then train 2k times and 1k times of iterations respectively. The trained SSD model is then used to initialize the DSSD network. The process of training MSSD is split into three steps. The first steps trains a primitive SSD model. The second stage: under such conditions, only the feature pyramid network module is trained, and the network parameters are not frozen, and the prediction model is added. Set the learning rate to $1 * e^3$, $1 * e^4$ to iterate 2k times and lk times respectively, in the third stage, the model is overall tuned.

3.3 Function Loss

To optimize the network for both class and bounding box localization, we are using the multi-task loss function. Let $x_{ij}^{p} = \{1,0\}$ be an indicator ground-truth variable for matching the i-th default box with the j-th ground truth box with p category. In our case the category of p can be a object or a background class. The matching variable x_{ij}^{p} is 1 when the IoU(equation 4.1) between the default bounding box and ground-truth is higher than 0.5. Furthermore, for each ground-truth bounding box, we also match the default-box with the highest IoU overlap. The value of x_{ij}^{p} is thus defined by:

$$x_j^p = \left\{ \begin{array}{l} j \ \ if \ \ 10U \geq 0.5 \, or \max 10U \\ 0 \ otherwise \end{array} \right\} \tag{3-4}$$

Additionally, because of the amount of default bounding boxes, the possibility also exist that more than one bounding box matches the ground truth. The matching of multiple bounding boxes strategy and the selecting bounding box with highest IoU overlap, are used to help the learning process with more positive samples to learn on the multi-task loss-function is definedas

$$\text{L(x,c,l,g)} = L_{conf}(x,c) + L_{loc}(x,l,g) \tag{3-5}$$

where the loss consists out of two task losses, the = $L_{conf}(x,c)$, which is the confidence andtheleaping boxregressionloss=$L_{conf}(x,l,c)$. Where$c$istheclassconfidence,lthe localization offset prediction, and g is the localization ground truth.

3.3.1 Localization Loss

In the localization loss, L_{loc}, a Huber loss is used,

$$L_\delta(d) = \left\{ \begin{array}{c} \frac{1}{2} d^2 \ \ For \ |d| \ \leq \delta \\ \delta \ \left(|d| - \frac{1}{2}\delta \right), otherwise \end{array} \right\} \tag{3-6}$$

Where d is representing the distance between the predicted localization and the ground-truth localization. If we set $\delta = 1$, we get the loss function which is known as the smooth L1-loss.

$$L1_{\delta}(d) = \begin{Bmatrix} 0.5d^2 \, for \, |d| \leq \delta \\ |d| - 0.5 \quad otherwise \end{Bmatrix} \tag{3-7}$$

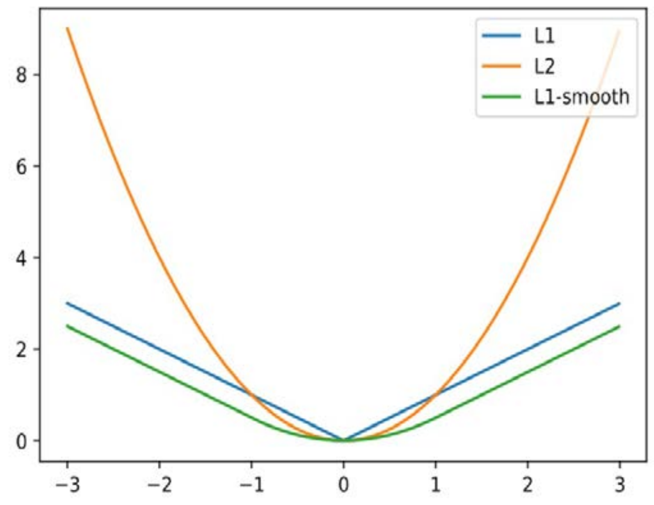


Figure3. 11: The L1, L2 and the L1s loss functions.

There are multiple reasons for using the $L1_g$ loss function graphically displayed in figure 3.11. Firstly, the loss function of the L1 is not differentiable at 0. Secondly, when

—d— ¡ 1 the loss function has a less steep gradient to better optimize towards the smaller distances. Thirdly, the gradient of the L2 becomes too large when the distance is large causing an unstable learning process, whereas the L1 loss function has a less hard constraint for points further away from the optimal position. The loss function between the predicted box $L_{l,o,c}$ is defined as followed,

$$L_{loc}(x, c^p, l_j, g_j) = \frac{1}{N\prime} \sum_{i\epsilon posmecx,cy,w,h}^{N\prime} \quad \sum x_{ij}^{p} L1_s(l_i^m - g'^{jm}) \tag{3-8}$$

Where $N^+ = \sum_{ij} x_{ij}^{p=1}$, which is a scalar for the amount of positive matches and l_i is

the localization prediction defined as the center off-set and the height and width off-set. In

the equation 3.8 d is replaced by $l_i^m - g_j^{nm}$ for g_i^{nm} a regression of the prediction center is made relative to its matched default bounding box and defined as followed,

$$g_j^{\prime\alpha} = (g_j^{\alpha x} - b_i^{\epsilon x})/b_i^{\vee} \tag{3-9}$$

$$g_j^{rcy} = (g_j^{cy} - b_i^{cy})/b_i^{h} \tag{3-10}$$

$$g_j^{nw} = \log\,(g_j^{w}/b_i^{w}) \tag{3-11}$$

$$g_j^{\wedge h} = \log\,(g_j^{h}/b_i^{h}) \tag{3-12}$$

The four coordinates of the ground truth are g^{cx}, g^{cy} for the center and g^{w}, g^{h} height and width. The $b_i^{w}, b_i^{h}, b_i^{cx}, b_i^{cy}$ respective coordinates of the matched default bounding box. Division of the height and width is used to normalize the width and the height. The log scale is used to balance the differences in scale, this makes the differences in small scale bounding boxes larger and larger differences for large bounding boxes smaller. The same is operations are done on the l_i^{m}

3.3.2 Confidence loss

The confidence loss, $L_{c,o,n,f}$ is a softmax function over the face class and background class denoted with P. Because of the large amount of default boxes the negative boxes greatly outnumber the positive bounding boxes. This creates a large class imbalance between back- ground (negative bounding boxes) and faces (positive bounding boxes), which makes the optimization process hard. To counter this issue, hard negative mining is used. Instead of summing over all the negative bounding boxes, the negative bounding boxes are sorted on class confidence and the top M negative bounding boxes are selected. Where the ratio between M and the positive bounding boxes is 3 : 1. The confidence loss is defined as followed,

$$L_{conf}(x,c) = -\tfrac{1}{N^{+}}\textstyle\sum_{i\epsilon portive}^{N'} x_{ij}^{p}\log(\hat{c}i^{p}) - \tfrac{1}{N'}\textstyle\sum_{i\epsilon Negative}\log\,(\hat{c}i^{0}) \tag{3-13}$$

$$where \qquad c_i^{\wedge p} = \frac{\exp\,(c_i^{p})}{\sum pexp(c_i^{p})} \qquad \text{and} \qquad N^{-} = M$$

3.4 Summary of this Chapter

This chapter first introduces the SSD model, and analyzes the characteristics and defects of the model. Then, based on the defects of the SSD model, a new model, MSSD model has proposed. After theoretical analysis and experiments, the MSSD model are still in the exposure correctness. The detection time is better than the traditional SSD model. However, since the number of prediction layer channels of the MSSD model based on ResNetl01 is too large, the network training speed is reduced. To avoid this situation, we should optimize the prediction layer channel. Moreover, we replace the up-sampling with the feature pyramid network in the MSSD text fusion. In order to improve the detection accuracy, we improved the feature pyramid network module, and borrowed the up-sampling method used in another context structure, namely TDM (Top-Down Modulation). Finally, we discussed the loss function used during training. The following chapter discusses the evaluation of the proposed object detection model based on the clinical measurements documented in this chapter.

CHAPTER 4

EXPERIMENT RESULT AND DISCUSSIONS

From the MSSD model introduced in the previous chapter, the research found that the main purpose of this chapter is to describe the experiments we perform to evaluate our models as well as discuss evaluation metrics that are used in evaluation. Moreover, we describe the data sets that are used for training and evaluation and any further implementation details

4.1 Datasets

The effect of traditional methods for object detection and recognition depends strongly on the characteristics of artificial selection of features. The effect of deep learning models on object detection and recognition has a great breakthrough compared with traditional technology, but it has a great effect on training the size of the data set that has larger requirements. Deep CNN's biggest challenge usually, the label of the image classification training is to collect examples. After all thousands of interpreted trainings to learn under supervision, in this part we will take a look at the datasets we've been working with. Since we focus mostly on deep convolutional neural networks, we will look at two datasets, which are VOC2007 and Logo dataset, for image classification.

4.1.1 Pascal VOC 2007/2012 datasets

Considering that there are many datasets available for training and validation, choosing the one that best serves our purposes is necessary and we selected the Pascal VOC dataset be- cause each class in contains many images along with bounding box data. Pascal VOC[40] was an annual challenge and workshop from 2005 to 2012, which has helped, promoted the development of image localization, classification and object detection. Five challenges were included: classification, detection, segmentation, action classification and person layout. One of the dataset used in this thesis is Pascal VOC 2007 / 2012 Challenge, that has 20 categories and includes 9,963 and 11,125 images respectively. The annotations of images rescue as XML files. In each of the files, the root nodes include folder, filename, source, size, object, and for each object, nodes include the name of its class, pose, truncated sign, difficulty and bounding box. Here the bounding boxes are annotated by the top-left and bottom-right corner of the objects, denoted as $[X_{max}, X_{min}, Y_{max}, X_{min}]$.

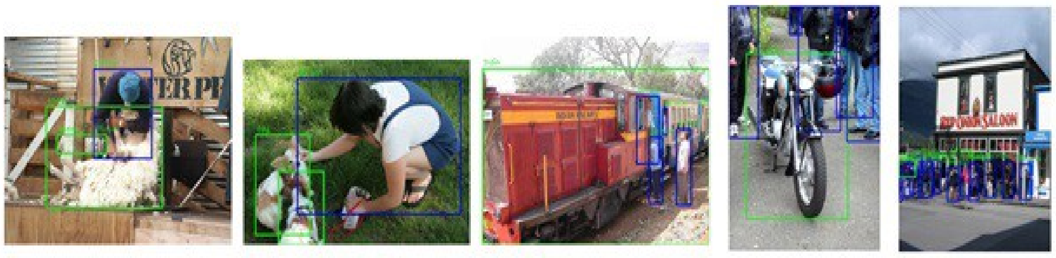


Figure4. 1: Examples of Pascal VOC ”person” dataset

4.1.2 Logo datasets

According to the investigation of the available datasets in the field of Logo detection and identification, it is found that the datasets involved in the current field are roughly BelgaL- ogos29, FlickrLogos-2730, FlickrLogos-32 / FlickrLogos-4731, and LOGO-NET [45]. In terms the size of data information, the LOGO-NET data collection is the most suitable data set for logo detection and recognition training, but its data set has not yet been open sourced. The open source data sets available in the field of logo detection and identification are relatively small in terms of the type of logo and the number of logos. Among them, the LOGO-NET data set proposed by Ali baba Group is the largest known data set in the field of logo detection and identification, but it is not yet open source, so the research in this field cannot use the LOGO-NET data set. Due to the limited means, we choose only eight classes from the

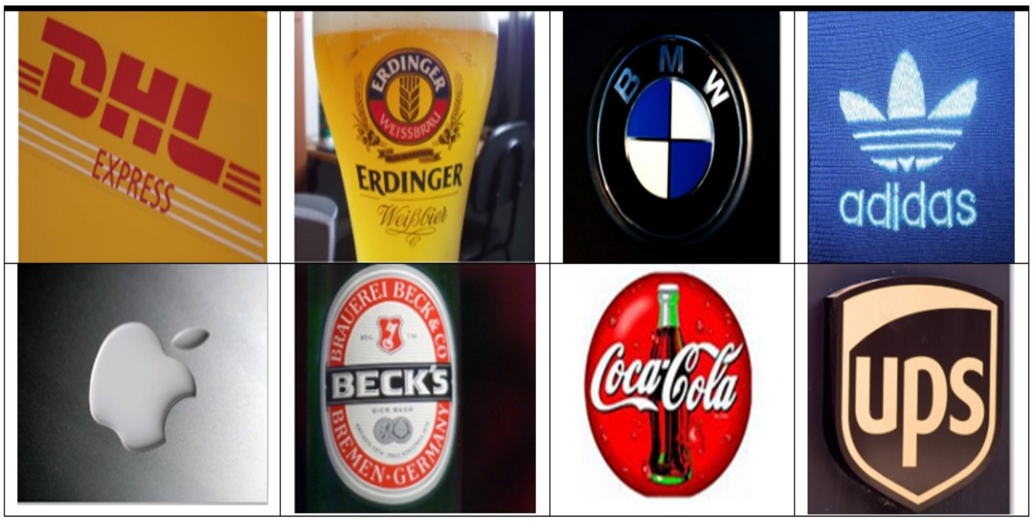


Figure4. 2: Eight different samples of Logo dataset

FlickrLogos-32 dataset namely, Adidas, Apple, Becks, BMW, Coca cola, Dhl, Erdinger and Ups, and increase the number of images from 70 to 220 in each class. To train and evaluate the MSSD object detection model, first the image and bounding box data sets were extracted for eight sections. One of the problems we've found with the derived datasets is that, the collected dataset consists of small sized images. We want to ensure, however, that the images was using to train and evaluate aren't tiny (i.e. at least 300 pixels).

- Gather Pictures: Object detection model requires hundreds of images to achieve a perfect classifier for detection. The training images should have random objects in the image along with the object objects, and should have a range of backgrounds and lighting conditions, in order to train a robust classifier. Some images must contain the desired object is partially ambiguous, overlaps with something else, or only half of the image opens. We have eight different objects which I want to identify for my Logo detection classifier (Adidas, Apple, Becks, BMW, Coca cola, Dhl, Erdinger and Ups). I got the images of my own dataset by download it from Google Image Search and pictures were carefully chosen with multiple objects overlapped in many images. The larger the pictures, the longer it takes for the classifier to train. So i made sure the images don't get too big and they should be under 200 KB. After gathering the images of dataset, i divided the data to 20% for testing and 80% for training from each class.

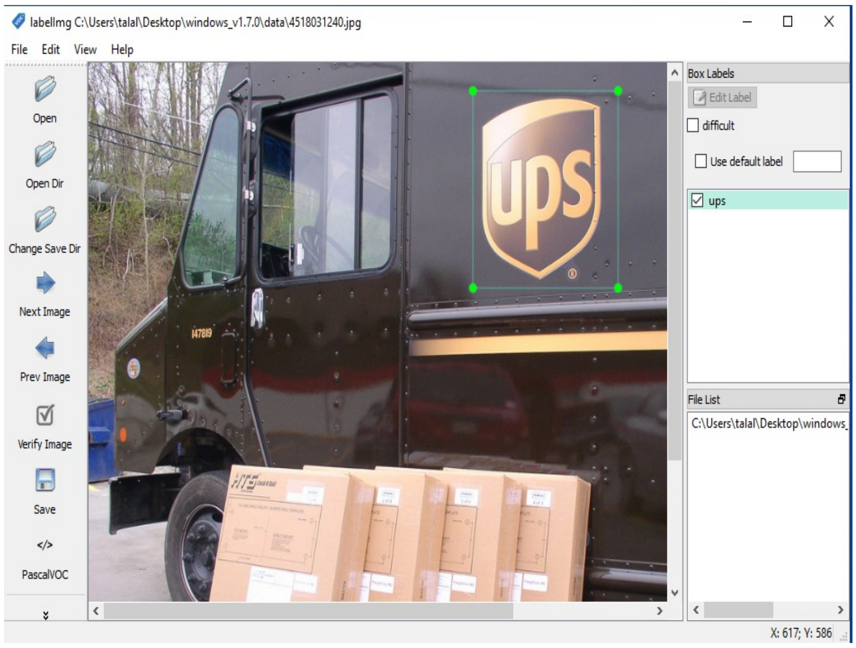


Figure4. 3: LabelIing saves a .xml file

- Label Pictures: Here's the exciting part! It's time to label the needed objects in each image with all the images collected. LabelImg is a great image labeling tool, and its GitHub page offers very simple instructions on how to download and use it https://github.com/tzutalin/labelImg. After downloading and installing LabelImg, drawabox aroundeachobjectineachimageandrepeattheprocessforalltheimages.

  LabelImg saves a file with a .xml extension for each image, containing the label data. Once each image has been labeled and saved, one.xml file will appear for each image.

4.2 System software and hardware construction

This section mainly introduces the construction of caffe framework related to deep learning experiments and the development of systems.

4.2.1 Construction of caffe framework

This paper chooses the caffe framework [41] for deep learning research and experiments although its building process is very cumbersome. The following briefly records the construction process.

- Ubuntu 16.04 installation: Ubuntu is one of the many versions of Linux. It is characterized by a friendly UI and a powerful package management mechanism. Ubuntu installation is relatively easy, download the corresponding one, burn the USB drive, restart the PC, enter the interface installation, and restart after the installation.
- Configure caffe:
  - Caffe relies on various library files, as well as python software.
  - Obtain caffe from github : git clone https://github.com/BVLC/caffe.
  - installation preparation, dependency installation : To installing, have a glance through this guide in https://caffe.berkeleyvision.org/installation.html
- install nvidia graphics driver : First go to Nvidia's official website (Fun Shark: Tour Diken: Man Q-type Dan Q base type i pendant: base object Saint 21 pad g three fortress: falling ink) to view the appropriate version, this experiment uses GTXl060 6G version graphics card. Corresponding graphics card, system type download corre- sponding driver.
- Install CUDA : CUDA (Compute Unified Device Architecture) is a computing platform launched by NVIDIA Graphics Corporation. CUDATM is a common purpose parallel computing architecture introduced by NVIDIA that has complex computing problems. You can use GPU to solve. In it CUDA is a parallel computing engine within the instruction set architecture (ISA) and GPU. In the scientific research community, CUDA is good the way

was made, and in 2009, deeply education ushers in the era of the GPU.

4.3 Evaluation matrics

In this section, we describe the evaluation metrics used to evaluate our models. The evaluation metrics used are precision recall curve and average precision.

4.3.1 Detection result

The evaluation of detection results requires a metric that determines whether a prediction is correct or not. The Intersection over Union(IoU) is a value used in object detection to measure the relevant predictions. To determine the IoU we need to have the bounding box ground truth B_{gt} and the bounding box prediction B_p. The IoU is defined as followed,

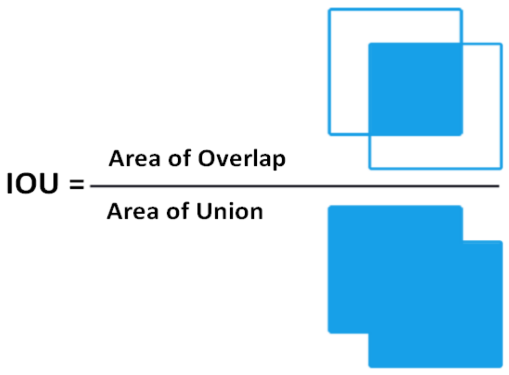


Figure4. 4: The IoU overlap graphically displayed by www.pyimagesearch.com

$$IOU = \frac{area(B_{gt} \cap B_p)}{area(B_{gt} \cup b_p)} \tag{4-1}$$

Since multiple detections on a single face will be counted as false positives, post-processing of the detections is required. The greedy non-maxima suppression as discussed previ- ously(chapter 3) reduces false positives prediction. All the remaining positive predictions are sorted by confidence. The highest positive prediction is considered a true positive (TP), the other predictions that an IoU $\leq$ 0.5 with the ground truth and have less scoreless are considered false positives (FP). The ground truth boxes that have no predictions assigned are considered false negatives (FN). True negatives (TN) are left out of consideration be- cause true negatives have no influence on the precision and recall.

4.3.2 PR curve explanation

With the definition of the relevant predictions described we can define the metric used to evaluate our models. Precision (P) is defined by how much of the prediction are correct, while recall (R) is defined by how many predictions are retrieved. Both P and R are defined as followed,

$$P = \frac{TP}{TP+FP} \tag{4-2}$$

$$R = \frac{TP}{TP+FN} \tag{4-3}$$

The precision and recall both show an important aspect of the retrieval performance of the model. Because precision and recall are inversely related, the trade-off between them is important. Moreover, the precision is usually computed at a certain cut-off. The cut-off both influences precision and recall, when the cut-off is higher it increases recall but decreases precision. Precision and recall with cut-off is defined by P(k) and R(k), where k is the cut-off at k bounding boxes. The trade-off between precision and recall can be combined into the precision-recall curve. The curve represents the precision and recall at different threshold values e.g., [0.1, 0.2 .... 0.9, 1]. At these threshold values the precision and recall is measured. To construct a smooth line the remaining points are interpolated.

4.3.3 Average precision

To further summarize the PR-curve into one metric, the area under the curve(AuC) can be computed. The AuC is the same as the average 28 precision and can be computed by taking the precision overall values of recall between 0 and 1,

$$\int_0^1 p(k)dk \tag{4-4}$$

The integral is an approximation and computed by the sum over precision at all different threshold values multiplied by the change in recall,

$$\frac{1}{P}\sum_{n=1}^{N} P(k)\, \Delta RW \tag{4-5}$$

where N is the total number of images in the dataset, k is the cut-off at k images and delta r is the change between R(k-1) and r(k). Instead of the average precision we use the interpolated average precision. The interpolated average precision replaces the precision at cut-off k by the maximum precision observed at all cut-offs with higher recall and is defined as followed,

$$\frac{1}{P}\sum_{h=1}^{N} \underset{\hat{k} > k}{max} p(\hat{k}) \vartriangle R(k) \tag{4-6}$$

4.4 Experimental result and analysis

The purpose of this thesis is to train the network and to accurately classify and locate the objects. The confidence of the prediction box is used to measure the correctness of the classification, while the coordinate information of the prediction frame measures the accuracy of the positioning. While the better object detection algorithm should have higher detection accuracy and detection speed, the detection accuracy is measured by mean Average Precision(mAP) and the detection speed is measured by FPS (Frames Per Second). In this part, this thesis will verify the accuracy of the object detection and the detection speed of the object detection. The MSSD model proposed in this thesis has better performance than the traditional SSD model on both datasets. In order to verify the effect of the feature pyramid network layer module and prediction model on detection performance, this thesis trained a VGG16 model with an input image 300*300 and a ResNetl01-SSD model with an input image of 321*321 for both Elt-sum and concat-sum models which lead to a slightly improve in accuracy at the expense of speed as you will see later. The source of the experimental data, i,e the source of the object detection image, is the VOC2007/2012 dataset and logo dataset.

4.4.1 Test results under VOC2007/2012 and logo dataset

Models have trained on the union of PASCAL VOC2007/2012 and Logo datasets which includes 20 categories and 8 classes respectively. The proposed MSSD model is implemented on the basis of the original SSD network built on the foundation of VGG16 and ResNet architectures and Caffe[41], all of which are available on the website. The baseline SSD has trained with a 300 x 300 input size and batch size of 16. Both of the two MSSD feature fusion models, concat-sum and Elt-sum, are fine-tuned on the well-trained SSD baseline for another 10K iterations. Learning rate is the amount of weights updated during training, so it is necessary to find a good value during training dataset. In our case changing the learning rate is a must and we gave 10^{-3} for the first 60K iterations and then reduce it to 10^{-4} and 10^{-5} at 70K iterations.

- Experiment of MSSD based on deep VGG16 network.

the appropriate layers are explored which will be the best to fuse with results of experimental, which had theoretically discussed in table 4.1. Yet we choose the two appropriate layers, conv4-3 and conv5-3, to fuse, because fc6 has a greater receptive field than conv5-3 for tiny objects that could add much more noise in the background. As it clear in table 4.1, the mAP of PASCAL VOC 2007 on general objects is taken into account. Additionally, when design- ing the modules we try different kernel numbers.

Table 4. 1: detection results of different fusion layers.

Layers	**Conv4-3**	**Conv4-3+Conv5-3**	**Conv4-3+fc6**	**Conv3-3+Conv4-3+Conv5-3**
Cocat	77.27	77.26	77.47	77.52
Eltsum	77.27	77.4	77.39	77.31

The two MSSD, concat-sum and Element-Sum, methods are both enhanced with regard to overall object detection compared to their SSD baseline. The MSSD with concat-sum module gets 77.6 mAP, while the element-sum module gets 77.4 mAP, which are 0.3 and 0.1 higher than the current SSD baseline.

Table 4. 2: Results of MSSD based on VGG16 network with IOU=0.5 on PASCAL VOC2007 test set

Method	**Network**	**mAP**	**aero**	**bike**	**bird**	**boat**	**bottle**	**bus**	**car**	**cat**	**chair**	**cow**
SSC300	VGG16	77.3	78.8	85.3	75.7	71.5	49.1	85.7	86.4	87.8	60.6	82.7
DSSD321	ResNet-101	78.6	81.9	84.9	80.5	68.4	53.9	85.6	86.2	88.9	61.1	83.5
Concat	VGG16	77.6	79.1	84.5	76.2	71.7	49.7	85.6	86.9	88.3	61.1	82.6
Elt_sum	VGG16	77.4	79.4	85.2	77.8	71.6	49.3	85.8	87.6	87.6	60.7	82.1
Method	**Network**	**mAP**	**aero**	**bike**	**bird**	**boat**	**bottle**	**bus**	**car**	**cat**	**chair**	**cow**
SSC300	VGG16	77.3	76.5	84.9	86.7	84	79.2	51.3	77.5	78.7	86.7	76.3
DSSD321	ResNet-101	78.6	78.7	86.7	88.7	86.7	79.7	51.7	78	80.9	87.2	79.4
Concat	VGG16	77.6	76.8	85.1	86.5	84.5	80.4	51.9	78.6	77.5	88.2	76.6
Elt_sum	VGG16	77.4	76.6	84.1	86.2	84.6	79.3	51.7	77.1	78.1	86.3	78

We test MSSD models on the logo dataset which contain 8 classes. The MSSD is implemented count on original SSD built on the VGG16 architectures and caffe framework. The MSSD with concat-sum module gets 77.0 mAP, while the element-sum module gets 76.7 mAP, which are 0.6 and 0.3 higher than the current SSD baseline. Since VOC2007

Table 4. 3: Results of MSSD based with IOU=0.5 on deep VGG16 network on Logo test set.

Method	Network	mAP	Average Precision in % for each class							
			Adidas	Apple	Becks	BMW	Coca-Cola	DHL	Erdinger	Ups
SSD300	VGG16	76.4	79.21	81.11	89.18	62.47	74.35	79.67	72.36	72.7
Concat	VGG16	77.0	81.4	81.3	89.9	63	75.4	79.2	72.6	73
Elt-sum	VGG16	76.7	81.1	79.6	89.1	61.9	74.9	79.8	73.12	74.3

dataset includes 20 classes and each class might have small objects, we choose 181 images,

Table 4. 4: Results of MSSD based on deep residual network with IOU=0.5 on Logo test set

Method	Network	mAP	Average Precision in % for each class							
			Adidas	Apple	Becks	BMW	Coca-Cola	DHL	Erdinger	Ups
SSD300	VGG16	76.7	78.1	82.4	88.7	68.7	73.9	78.18	72.63	71.14
Elt-sum	ResNet-101	76.9	79.6	82.1	89.1	69.4	72.9	78.8	73.12	71.5

As can be seen from table 4.5, the MSSD achieved improvement compared with the traditional SSD model. Since our logo dataset contains multiple overlapped objects and small objects in many images, you may find an enhancement compared to the current SSD model in terms of the concat-sum and element-sum modules. Results for the detection are shown in Fig 4.5. We found that small objects, with particular background, detection performance slightly improved. And more accurately detected the objects that often appear along with relative objects.

4.5 Result Evaluation

All experiments will be on top of PASCAL VOC2007 dataset and Logo dataset. All experiments will be on top of PASCAL VOC2007 dataset and Logo dataset. The results of the system's evaluation in terms of its ability to detect and recognize the images contained in the test set will then be presented, these results being presented using a plot of precision / recall curve and average precision metric. In 4.3, we implement how to measure the recall and precision.

4.5.1 Result Analysis on PASCAL VOC2007 datasets

We addressed the datasets we use for experiments in 4.1, and it is also clear that video stream data show differences in aspect ratio, resolution, motion blur, lighting etc. In this section, we evaluate the generalization ability on PASCAL VOC2007 dataset. Figure 4.5 shows the evaluation results of 3 models-SSD300, MSSD with concate-sum and concate-sum methods.

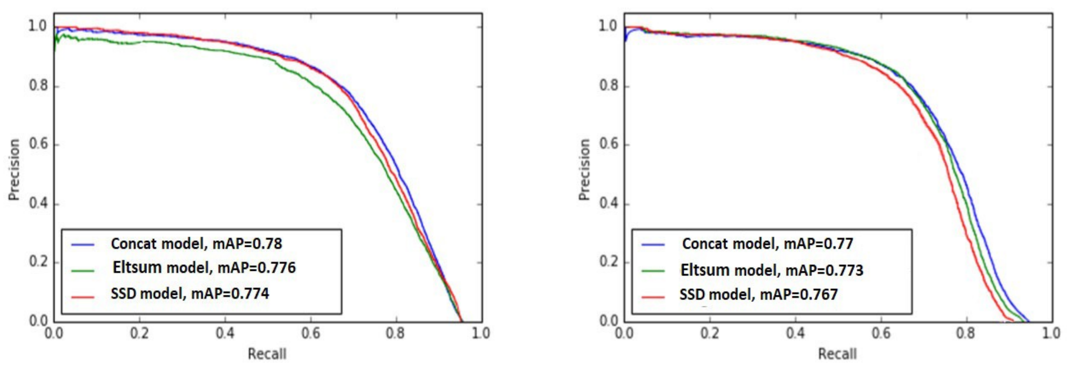


Figure4. 5: The precision-recall curves of SSD and MSSD models.

The results on Pascal VOC 2007 differ slightly from the precision reported in the original papers. It may be induced by various ways of implementation.

As shown in Figure 4.6, because we began our training from a pre-trained checkpoint instead of starting from scratch, the total loss value decreases rapidly. The total loss values ‘uctuate but exhibit reducing behavior overall.

Figure 4.7 demonstrates overall mAP development for 80,000 steps at 0.5 IoU. The mAP values are evaluated for the validation dataset at 0.5 IoU. As is evident from Figure 4.7, in around 6,800 measures, mAP is witnessing a tremendous increase to 67.8 percent. The mAP then increases slightly more and in 19,000 steps reaches closer to 77.6 per cent. The mAP value remains relatively constant with minor fluctuations after 19,000 moves.

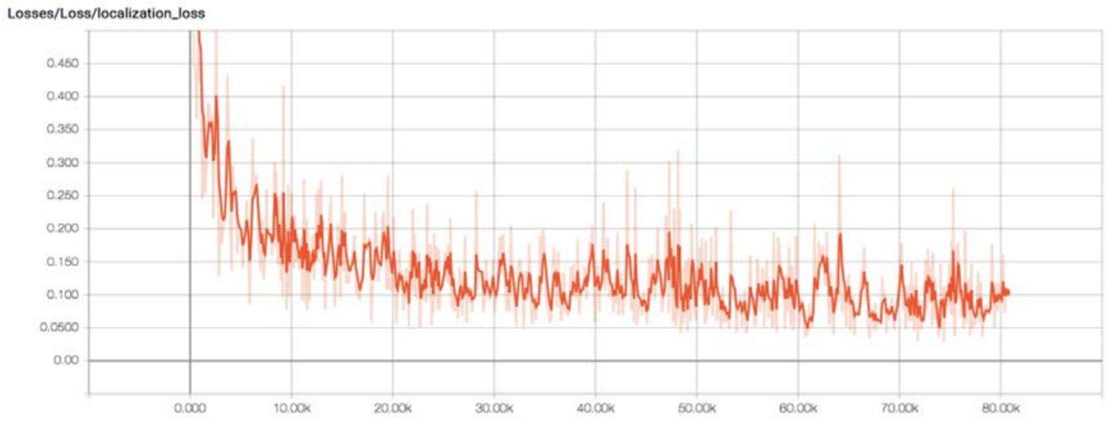


Figure4. 6: Decline of total loss when concate-sum on PASCAL VOC2007 dataset

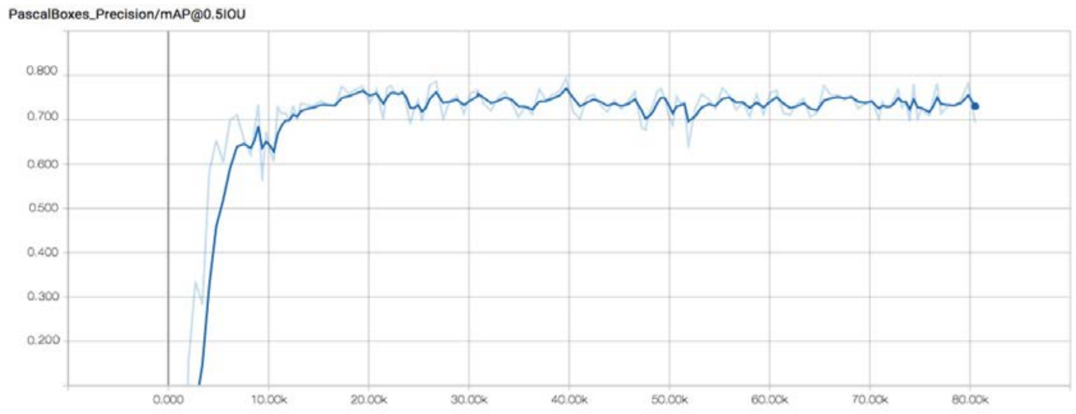


Figure4. 7: Development of overall mAP when concate-sum on PASCAL VOC2007 dataset.

Because the mAP values leveled out after 19,000 moves, the model is exported at various training stages and checked on the test dataset.

4.5.2 Result Analysis on logo dataset

The MSSD model is fine-tuned for 100,000 steps as shown in Figures 4.8 and 4.9. Figure

4.9 demonstrates the downward trend in total loss during the entire training phase. For each iteration the total loss can differ slightly from the previous iteration. The key point, however, is that overall loss values decreased during the training course.

Figure 4.9 shows overall mAP development at 0.5 IoU per 100,000 steps. For the validation dataset the mAP values are evaluated at 0.5 IoU. As is evident from Figure 4.9, in around 9000 stages, mAP is experiencing a tremendous increase up to 53% percent. The mAP then increases slightly more and in 33,000 steps reaches nearer to 73%. The mAP

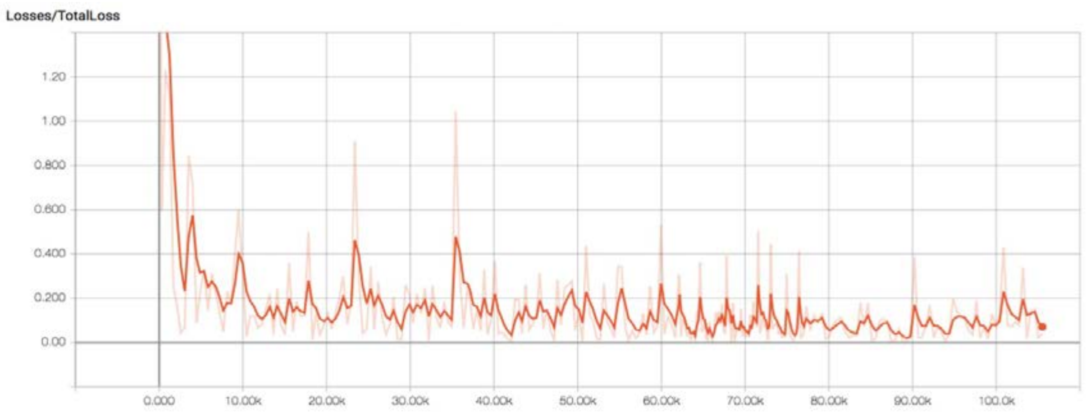


Figure4. 8: Decline of total loss when Eltsum-ResNet-101 on PASCAL VOC2007 dataset.

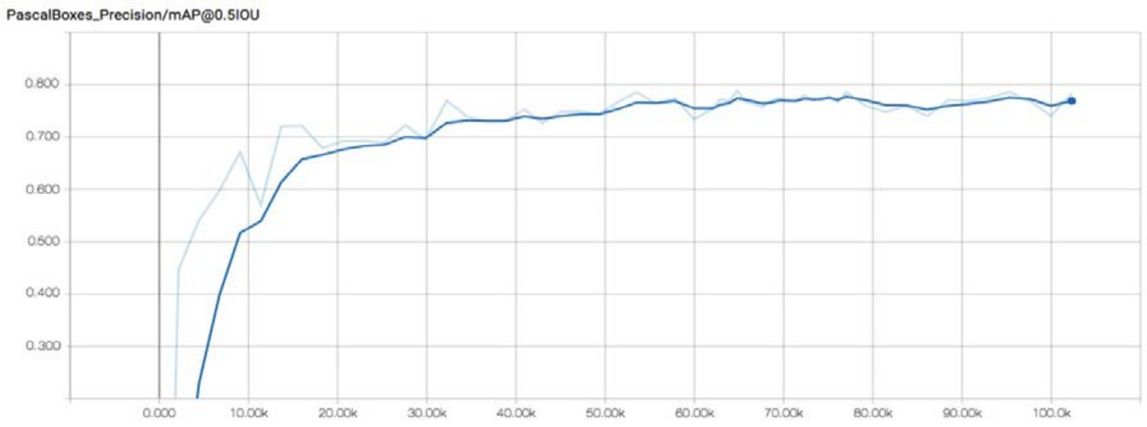


Figure4. 9: Development of overall mAP when Eltsum-ResNet-101 on PASCAL VOC2007 dataset.

value increased little after 19,000 measures and settled down with minor fluctuations of 77.4 percent.

4.6 Running Time

On PASCAL VOC 2007 and Logo test datasets, running time for those both fusion methods has evaluated, as can be seen in Table 4.6. The two fusion methods, concat-sum and element-sum modules, have detection speed 40 FPS and 43 FPS respectively in PASCAL VOC 2007 test dataset, while it is 13.12 FPS in Logo test dataset. Unfortunately because of the additional feature fusion layers, both became slower than original SSD model. Nevertheless MSSD fusion methods are also still achieving a real-time detection. The element-sum

Table 4. 5: The running time illustration of different models.

Dataset	Method	Network	mAP	FPS
PASCAL VOC 2007 Test	SSD300	VGG16	77.3	50
	DSSD321	Residual-101[11]	78.6	13.6
	MSSD Concat Model	VGG16	77.6	40
	MSSD Eltsum Model	VGG16	77.4	43
	MSSD Eltsum Model	Residual-101[11]	77.4	13.12
Logo Test	SSD300	VGG16	76.4	50
	MSSD Concat Model	VGG16	77	40
	MSSD Eltsum Model	VGG16	76.7	43
	MSSD Eltsum Model	Residual-101[11]	76.9	13.12

module has used 2 convolution layers in each layer, with 384 kernels. and the concat-sum model used 3 convolution layers in each layer with 512 kernels. Which is why the model with the element-sum approach is faster than the model with the concat-sum approach by 3 FPS. The decreasing of number of frame per second due to the extra operations taken by MSSD model, this decreasing beside improving the slight increasing of accuracy will be solved in my future studying.

4.7 Performance comparison of MSSD Modules

4.7.1 Performance of MSSD modules based on VGG16 Network

Here we show a comparison in performance of the pre-trained SSD300 model and the fine- tuned MSSD models on both datasets, the experimental results are demonstrate that helpful contextual information proves the existence of small objects.

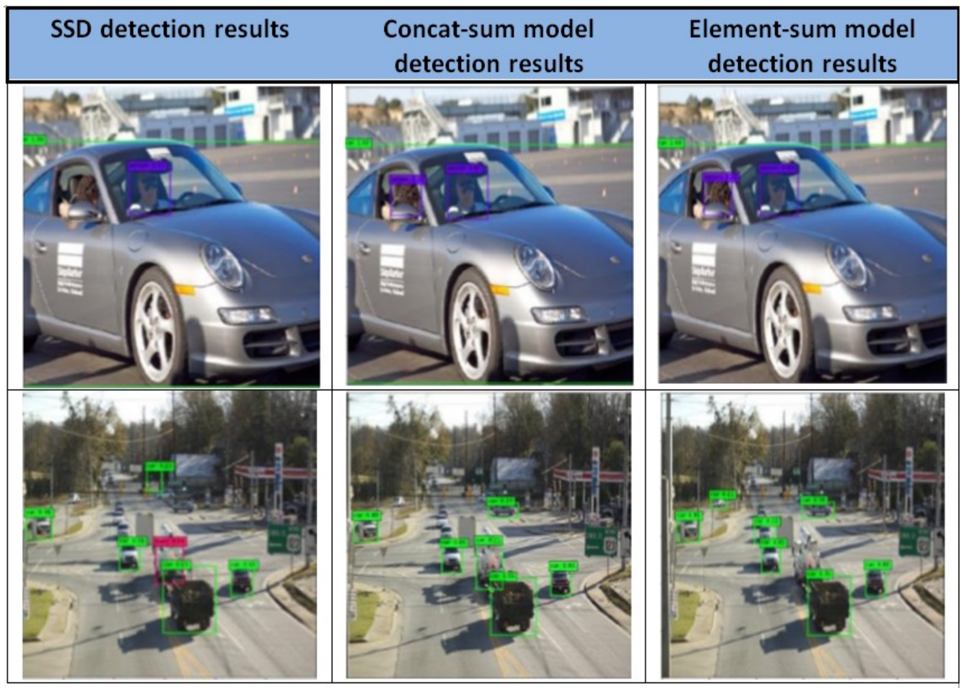

SSD detection results
Concat-sum model detection results
Element-sum model detection results

Figure4. 10: Show the detection results original SSD and MSSD model with concat-sum and element-sum module, respectively on Pascal VOC 2007 dataset.

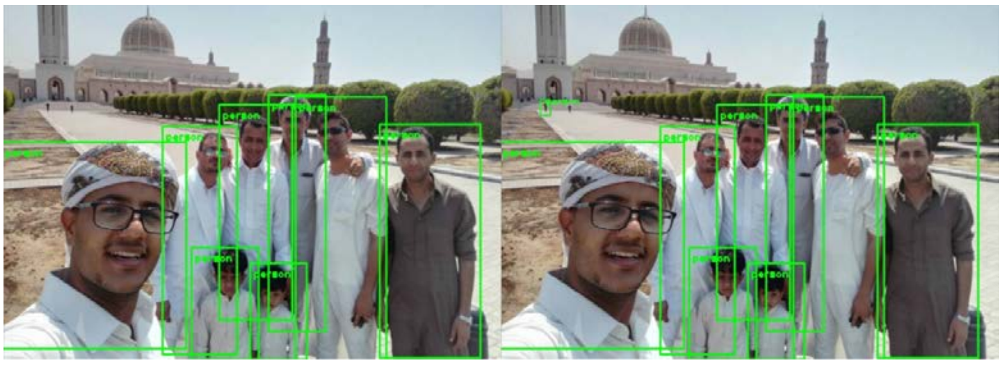


Figure4. 11. Right: The results of concat-sum model detection. Left: The results of element-sum model detection. Concat-sum fusion model could weaken the noise of background interference while element- sum fusion model can't weaken. When looking into the difference of these MSSD methods, we carefully analyze the fusion methods and the results of their detection. In Fig 4.11, which does not include im- portant contextual information. Although the concat-sum module utilizes learned weights to merge object feature with context feature, it can then choose useful contextual informa- tion and diminishes background noise interference. Unfortunately, the element-sum method marge both object features and context in an equivalent way, therefore it cannot adapt the beneficial contextual information. Conversely, in Fig 4.12, The cars in the scene are blurred, so the context is important for identification. In this situation, the element-sum fusion ap- proach performs better than the concat-sum fusion approach, since the latter seems to have more choice that perhaps the bond between object and context might not be well learned.

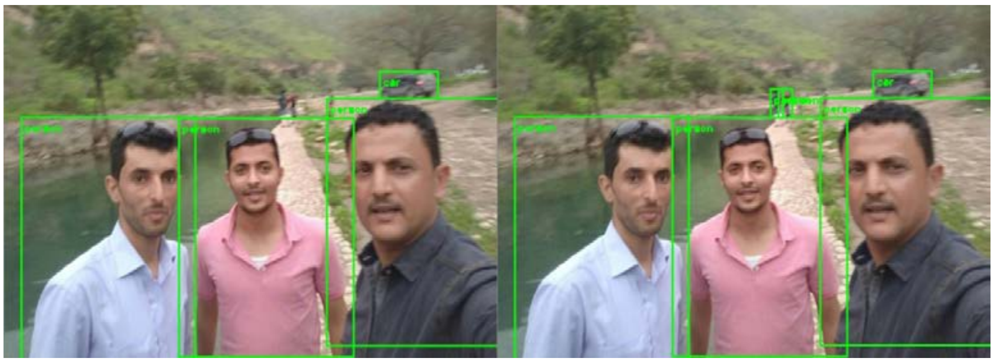


Figure4. 12. Right: The effects of element-sum fusion model detection. Left: The results of concat- sum fusion model detection. The children in this picture are small and blurred, so that the contextual information is required to identify them. This context is exploited well enough by the element-sum fusion model, whereas the concat-sum fusion model cannot.

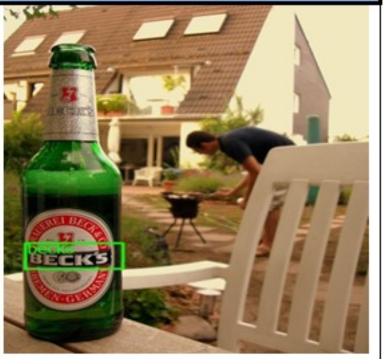
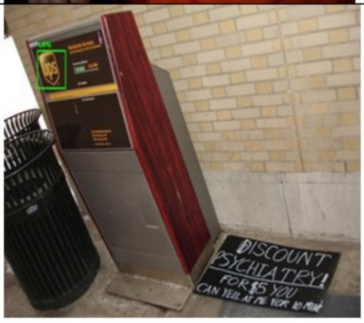
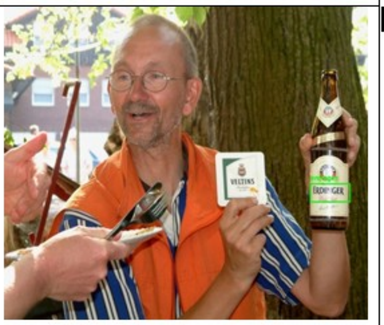
Data set	concat-sum model detection results	Element-sum model detection results
Pascal VOC 2007		
Logo		

Figure4. 13: Show the detection results original SSD and MSSD model with concat-sum and element-sum module, respectively on logo dataset.

In fig 4.13 we show results of detection of the MSSD models on Logo dataset, the experimental results are demonstrate that helpful contextual information proves the existence of small objects.

4.7.2 Performance of SDD Modules based on Residual Network

The reason why the better detection result cannot be obtained is that the SSD model itself is based on a deep convolutional neural network, and the high-level feature information with a large receptive field is used to predict a large object, and the low-level feature information with a small receptive field is used to predict a small object. When the object is lacking, the SSD is less effective for detecting small objects due to the lack of high-level semantic features. The MSSD model uses a context-based fusion method. The experimental results are shown in Fig.4.14 below.

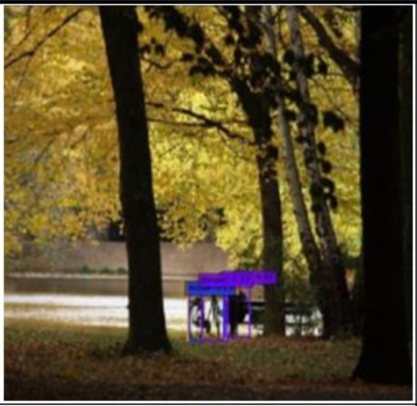
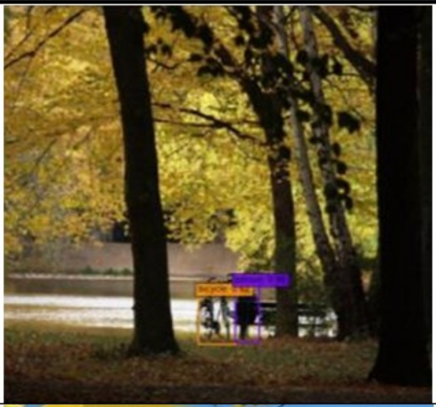
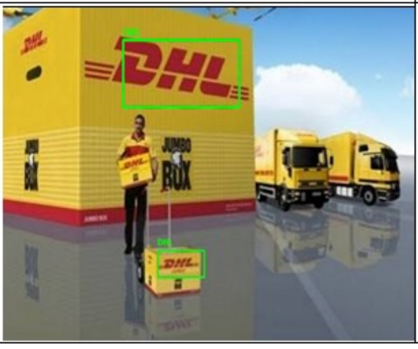
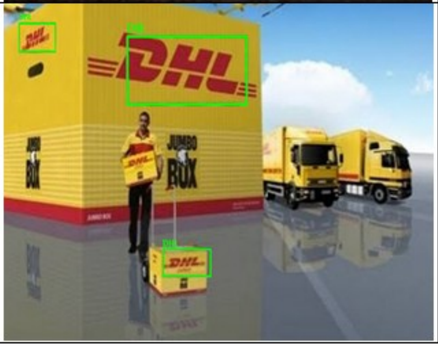
data set	SSD detection results	Element-sum model detection results
Pascal VOC 2007		
Logo		

Figure4. 14: The results of traditional SSD model and MSSD model.

The left side of the figure shows the detection results of the original SSD model, and the right side is the final result of the MSSD model object detection. It can be clearly seen from Fig 4.13 and Fig 4.14 that the traditional SSD model cannot detect the object in the image more accurately, and the MSSD can detect more objects. For the objects with similar categories, the detection result of the MSSD is accurate.

4.8 Summary of this Chapter

In this chapter, we discussed how we collect the Logo dataset beside Pascal VOC 2007/2012 dataset. We discussed the evaluation of the MAP-based object detection model. We have pre-trained SSD Discuss the performance difference between the model and the fine MSDS model in the test data set. We have MSSD based on speed and accuracy the fine tune model is comparable and analyzed. To concluded, theoretical analysis and experiments have proved that the MSSD model is better than the MSSD model in terms of detection accuracy but not detection time

CHAPTER 5

CONCLUSION

This chapter summarizes this thesis, discusses the findings and ends with the recommendations on future work to enhance the detection of small objects.

5.1 Summary of work

This thesis mainly studies the method of object detection based on SSD model. An improved SSD object detection algorithm MSSD is proposed. In this thesis, the context-based network structure is used to fuse the upper and lower layers and the traditional up-sampling structure is improved. The high-level semantic information is embedded in the low-level network's feature information, and the multi-scale feature map of the prediction regression position box and the classification task input is enriched to improve the detection accuracy. The VGG16 network used for SSD training was used with a deep residual network to optimize the feature maps of candidate box regression and classification task input, while showing the experimental results and corresponding advantages and disadvantages. In order to solve the problem of SSD model, an improved SSD model, namely MSSD model, is introduced in detail, and the problems existing in SSD model are mainly analyzed. This thesis uses the FPN-based network structure to fuse the upper and lower layers and improves the traditional up-sampling structure. The high-level semantic information is embedded in the low-level network's feature information, and the multi-scale feature map of the prediction regression position box and the classification task input is enriched to improve the detection accuracy. Compared to the state of the art Small Object Detector, experiments show that the MSSD model is better than the traditional SSD model in detection accuracy but not detection speed.

In conclusion we observe that although the SSD framework is scale-invariant it can still benefit from the feature fusion architecture to detect objects of different sizes. The SSD method together with the feature fusion architecture can be adapted to work for the object detection task.

5.2 Future work

There are many problems in object detection, (1) multi-feature fusion of context features, multi-scale object localization, high-level feature information with wider receptive fields and multi-scale maps for object prediction for predicting large objects. Low-level receptive field information predicts small objects. (2) It is difficult to meet real-time performance for high-resolution images or videos. A object with a simple background, sufficient lighting, no obstruction, and a shooting angle of front is relatively easy to detect. When the background is mixed with the object, there are obstructions near the object, the light intensity is too weak, and the object pose changes The rate is greatly reduced. Although the two algorithms proposed in this thesis have achieved a better improvement in detection accuracy than the previous algorithms, there are still some problems to be solved. There are other ways for high-level and low-level feature fusion, so the model should continue to be optimized to find more effective methods with contextual features to enable it to obtain higher accuracy. Because of the above difficulties, finding an ideal object detection algorithm still needs continuous research and improvement. Object detection has important practicability in artificial intelligence, autonomous driving, smart transport, face recognition and other fields but it also has great prospects and that will be the further work to study.

REFERENCES

OUYANG W, WANG X (2013). Joint deep learning for pedestrian detection. *In proceeding of IEEE International Conference on Computer Vision,* 2056 – 2063.

STENROOS O, OTHERS (2017). Object detection from images using convolutional neural networks.

VIOLA P, JONES M(2001).Rapid object detection using a boosted cascade of simple features. *In Proceeding of IEEE computer society conference on computer vision and pattern recognition,* CVPR Vol1,I – I.

HASTIE T, ROSSET S, ZHU J,(Lea et al 2013).Multi-class adaboost. Volume 349–360.

RUSSAKOVSKY O, DENG J, SU H, (et al. 2015). Image-net large scale visual recognition challenge *In 2015 International journal of computer vision 115(3).* 211 – 252.

GIRSHICK R, DONAHUE J, DARRELL T, (et al. 2014). Rich feature hierarchies for accurate object detection and semantic segmentation .*In Proceeding of IEEE conference on computer vision and pattern recognition.* 580 – 587

HE K, ZHANG X, REN S, (Lea et al. 2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *In proceeding of IEEE transactions on pattern analysis and machine intelligence, 37(9),*1904 – 1916

GIRSHICK R (2015). Fast R-CNN. *In proceeding of IEEE international conference on computer vision*, 1440 – 1448

WU J, REHG J M. Centrist (2010). A visual descriptor for scene categorization[J]. *In proceeding of IEEE transactions on pattern analysis and machine intelligence, 33(8),* 1489 – 1501

LIU W, ANGUELOV D, ERHAN D, (Lea et al. 2016). SSD Single shot multi-box detector. *In 2016 European conference on computer vision.* 21 – 37.

CAO G, XIE X, YANG W,(Lea et al. 2017). Feature-fused SSD: Fast detection for small objects. *In 2017 Ninth International Conference on Graphic and Image Processing (ICGIP). Vol 10615 2018 106151E.*

EGGERT C, ZECHA D, BREHM S, (Lea et al. 2017). Improving small object proposals for company logo detection. *In 2017 ACM on International Conference on Multimedia Retrieval*. 167 – 174

WILMS C, FRINTROP S.(2018). Attention Mask, Attentive, Efficient Object Proposal Generation Focusing on Small Objects. *In 2018 Asian Conference on Computer Vision*. 678 – 694.

HE K, ZHANG X, REN S, (Lea et al.(2016). Deep residual learning for image recognition. *In proceeding of IEEE conference on computer vision and pattern recognition.* 770 – 778

EVERINGHAM M, WINN J.(2012). The PASCAL visual object classes challenge 2012 (VOC2012) development kit. Pattern Analysis, Statistical Modelling and Computational Learning, Tech, Rep.

GOODFELLOW I, BENGIO Y, COURVILLE A(2016). Deep learning, [M]. [S.l.] MIT press.

IOFFE S, SZEGEDY C(2015). Batch normalization Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167.

EGGERT C, BREHM S, WINSCHEL A, (Lea et al.2017). A closer look: Small object detection in faster R- CNN[. *In proceeding of IEEE international conference on multimedia and expo(ICME).* 421 – 426.

KRISHNA H, JAWAHAR C(2017). Improving small object detection, *In 2017 4th IAPR Asian Conference on Pattern Recognition(ACPR).* 340 – 345.

REDMON J, DIVVALA S, GIRSHICK R, (Lea et al.2016). You only look once: Unified, real-time object detection. *In proceeding of IEEE conference on computer vision and pattern recognition,* 779 – 788

REN S, HE K, GIRSHICK R, (Lea et al.2015). Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in neural information processing systems: 91–99.

KRIZHEVSKY A, SUTSKEVER I, HINTON G E (2012). Image-net classification with deep convolutional neural networks, Advances in neural information processing systems: 1097–1105.

LIU L, OUYANG W, WANG X, (Lea et al.2018) Deep learning for generic object detection: A survey. *In 2018 International Journal of Computer Vision*, 1809: 1 – 58

SERMANET P, EIGEN D, ZHANG X, (et al.2013). Over feat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

SIMONYAN K, ZISSERMAN A (2014). Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556,

HINTON G E, SALAKHUTDINOV R R (2006). Reducing the dimensionality of data with neural networks science, 313(5786): 504–507.

LECUN Y, BENGIO Y, HINTON G (2015). Deep learning nature, 521(7553): 436 – 444.

LAW H, DENG J(2018). Corner-net: Detecting objects as paired key-points. *In 2018 European Conference on Computer Vision (ECCV):* 734–750.

FUKUSHIMA K (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks, 1(2): 119 – 130.

HERNÁNDEZ D C, FILONENKO A, SHAHBAZ A, (et al.2017). Lane marking detection using image features and line fitting model. *In 2017 10th International Conference on Human System Interactions (HSI):* 234 – 238.

ZHAO Q, SHENG T, WANG Y, (et al.2018). Cfenet: An accurate and efficient single-shot object detector for autonomous driving. arXiv preprint arXiv:1806.09790.

FU C-Y, LIU W, RANGA A, (et al.2017) DSSD: Deconvolutional single shot detector[J]. arXiv preprint arXiv:1701.06659.

LECUN Y, BOSER B, DENKER J S, (et al.1989) Back propagation applied to handwritten zip code recognition. Neural computation, 1(4): 541 – 551.

LIU M, DONG J, DONG X, (et al.2018). Segmentation of lung nodule in CT images based on mask R-CNN. *In 2018 9th International Conference on Awareness Science and Technology (iCAST):* 1 – 6.

LI J, LIANG X, WEI Y, (et al.2017). Perceptual generative adversarial networks for small object detection. *In proceeding of IEEE conference on computer vision and pattern recognition:* 1222 – 1230.

BELL S, LAWRENCE ZITNICK C, BALA K, (et al.2016). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *In proceeding of IEEE conference on computer vision and pattern recognition:* 2874 – 2883.

HINTON G E, SRIVASTAVA N, KRIZHEVSKY A, (et al.2012). Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580.

SRIVASTAVA N, HINTON G, KRIZHEVSKY A, (Lea et al.2014). Dropout: a simple way to prevent neural networks from over fitting. *The journal of machine learning research, 15(1)*: 1929 – 1958.

SRIVASTAVA N (2013). Improving neural networks with dropout, University of Toronto, 182(566): 7.

EVERINGHAM M, VAN GOOL L, WILLIAMS C K, (Lea et al.2010). The pascal visual object classes (voc) challenge. *In 2010 International journal of computer vision, 88(2):* 303 – 338.

JIA Y, SHELHAMER E, DONAHUE J, (Lea et al.2014). Caffe: Convolutional architecture for fast feature embedding, *In 2014) 22nd ACM international conference on Multimedia:* 675 – 678.

HAN J, MORAGA C (1995). The influence of the sigmoid function parameters on the speed of backprop- agation learning, *in 1995 International Workshop on Artificial Neural Networks:* 195 – 201.

NAIR V, HINTON G E (2010). Rectified linear units improve restricted boltzmann machines, *In 2010 27th international conference on machine learning (ICML-10):* 807 – 814.

UIJLINGS J R, VAN DE SANDE K E, GEVERS T, (Lea et al.2013). Selective search for object recognition, In *2013 International journal of computer vision 104(2)*: 154 – 171.

YANG B, YAN J, LEI Z, (Lea et al.2016). Craft objects from images, *In 2016 IEEE Conference*

on computer vision and pattern recognition: 6043 – 6051.

ZENG X, OUYANG W, YAN J, (Lea et al.2017). Crafting gbd-net for object detection, *In proceeding of IEEE transactions on pattern analysis and machine intelligence 40(9)*: 2109 – 2123.

HEARST M A, DUMAIS S T, OSUNA E, (Lea et al.1988). Support vector machines, *In proceeding IEEE Intelligent Systems and their applications, 13(4)* : 18 – 28.

APPENDICES

Appendix A : SSD Architecture based on vgg16 Model

```
def __init__(self, phase, size, base, extras, head, num_classes):
        super(SSD, self).__init__()
        self.phase = phase
        self.num_classes = num_classes
        self.cfg = (coco, voc)[num_classes == 21]
        self.priorbox = PriorBox(self.cfg)
        self.priors = Variable(self.priorbox.forward(), volatile=True)
        self.size = size
        # SSD network
        self.vgg = nn.ModuleList(base)
        # Layer learns to scale the l2 normalized features from conv4_3
        self.L2Norm = L2Norm(512, 20)
        self.extras = nn.ModuleList(extras)

        self.loc = nn.ModuleList(head[0])
        self.conf = nn.ModuleList(head[1])
        if phase == 'test':
            self.softmax = nn.Softmax(dim=-1)
            self.detect = Detect(num_classes, 0, 200, 0.01, 0.45)
    def forward(self, x):
        """Applies network layers and ops on input image(s) x.
        Args:
```

```
    x: input image or batch of images. Shape: [batch,3,300,300].
Return:
    Depending on phase:
    test:
        Variable(tensor) of output class label predictions,
        confidence score, and corresponding location predictions for
        each object detected. Shape: [batch,topk,7]
    train:
        list of concat outputs from:
            1: confidence layers, Shape: [batch*num_priors,num_classes]
            2: localization layers, Shape: [batch,num_priors*4]
            3: priorbox layers, Shape: [2,num_priors*4]
sources = list()
loc = list()
conf = list()
# apply vgg up to conv4_3 relu
for k in range(23):
    x = self.vgg[k](x)
s = self.L2Norm(x)
sources.append(s)
# apply vgg up to fc7
for k in range(23, len(self.vgg)):
    x = self.vgg[k](x)
```

```
        sources.append(x)
```

Appendix B: Modified Single Shot Multi-box Detector

```
        for k, v in enumerate(self.extras):
            x = F.relu(v(x), inplace=True)
            if k % 2 == 1:
                sources.append(x)
        # apply multibox head to source layers
        for (x, l, c) in zip(sources, self.loc, self.conf):
            loc.append(l(x).permute(0, 2, 3, 1).contiguous())
            conf.append(c(x).permute(0, 2, 3, 1).contiguous())
        loc = torch.cat([o.view(o.size(0), -1) for o in loc], 1)
        conf = torch.cat([o.view(o.size(0), -1) for o in conf], 1)
        if self.phase == "test":
            output = self.detect(
                loc.view(loc.size(0), -1, 4),                   # loc preds
                self.softmax(conf.view(conf.size(0), -1,
                                       self.num_classes)),      # conf preds
                self.priors.type(type(x.data))                  # default boxes
            )
        else:
            output = (
                loc.view(loc.size(0), -1, 4),
                conf.view(conf.size(0), -1, self.num_classes),
                self.priors
            )
        return output
    def load_weights(self, base_file):
        other, ext = os.path.splitext(base_file)
        if ext == '.pkl' or '.pth':
            print('Loading weights into state dict...')
            self.load_state_dict(torch.load(base_file,
                                map_location=lambda storage, loc: storage))
            print('Finished!')
        else:
            print('Sorry only .pth and .pkl files supported.')
# This function is derived from torchvision VGG make_layers()
def vgg(cfg, i, batch_norm=False):
    layers = []
    in_channels = i
    for v in cfg:
        if v == 'M':
            layers += [nn.MaxPool2d(kernel_size=2, stride=2)]
        elif v == 'C':
            layers += [nn.MaxPool2d(kernel_size=2, stride=2, ceil_mode=True)]
        else:
            conv2d = nn.Conv2d(in_channels, v, kernel_size=3, padding=1)
            if batch_norm:
                layers += [conv2d, nn.BatchNorm2d(v), nn.ReLU(inplace=True)]
            else:
                layers += [conv2d, nn.ReLU(inplace=True)]
            in_channels = v
    pool5 = nn.MaxPool2d(kernel_size=3, stride=1, padding=1)
    conv6 = nn.Conv2d(512, 1024, kernel_size=3, padding=6, dilation=6)
    conv7 = nn.Conv2d(1024, 1024, kernel_size=1)
    layers += [pool5, conv6,
               nn.ReLU(inplace=True), conv7, nn.ReLU(inplace=True)]
    return layers
def add_extras(cfg, i, batch_norm=False):
    # Extra layers added to VGG for feature scaling
    layers = []
    in_channels = i
    flag = False
    for k, v in enumerate(cfg):
        if in_channels != 'S':
            if v == 'S':
                layers += [nn.Conv2d(in_channels, cfg[k + 1],
                                     kernel_size=(1, 3)[flag], stride=2, padding=1)]
            else:
                layers += [nn.Conv2d(in_channels, v, kernel_size=(1, 3)[flag])]
            flag = not flag
        in_channels = v
    return layers
def multibox(vgg, extra_layers, cfg, num_classes):
    loc_layers = []
    conf_layers = []
    vgg_source = [21, -2]
    for k, v in enumerate(vgg_source):
        loc_layers += [nn.Conv2d(vgg[v].out_channels,
                                 cfg[k] * 4, kernel_size=3, padding=1)]
        conf_layers += [nn.Conv2d(vgg[v].out_channels,
                                  cfg[k] * num_classes, kernel_size=3, padding=1)]
    for k, v in enumerate(extra_layers[1::2], 2):
        loc_layers += [nn.Conv2d(v.out_channels, cfg[k]
                                 * 4, kernel_size=3, padding=1)]
        conf_layers += [nn.Conv2d(v.out_channels, cfg[k]
                                  * num_classes, kernel_size=3, padding=1)]
    return vgg, extra_layers, (loc_layers, conf_layers)


base = {
    '300': [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 'C', 512, 512, 512, 'M',
            512, 512, 512],
    '512': [],
}
extras = {
    '300': [256, 'S', 512, 128, 'S', 256, 128, 256, 128, 256],
    '512': [],
}
mbox = {
    '300': [4, 6, 6, 6, 4, 4],  # number of boxes per feature map location
    '512': [],
}
def build_ssd(phase, size=300, num_classes=21):
    if phase != "test" and phase != "train":
        print("ERROR: Phase: " + phase + " not recognized")
        return
    if size != 300:
        print("ERROR: You specified size " + repr(size) + ". However, " +
              "currently only SSD300 (size=300) is supported!")
        return
    base_, extras_, head_ = multibox(vgg(base[str(size)], 3),
```

Appendix C: Modified Single Shot Multi-box Detector based on Residual Network

```
def resnet_v1_101(inputs,
                  num_classes=None,
                  is_training=True,
                  global_pool=True,
                  output_stride=None,
                  spatial_squeeze=True,
                  store_non_strided_activations=False,
                  min_base_depth=8,
                  depth_multiplier=1,
                  reuse=None,
                  scope='resnet_v1_101'):
  """ResNet-101 model of [1]. See resnet_v1() for arg and return description."""
  depth_func = lambda d: max(int(d * depth_multiplier), min_base_depth)
  blocks = [
      resnet_v1_block('block1', base_depth=depth_func(64), num_units=3,
                      stride=2),
      resnet_v1_block('block2', base_depth=depth_func(128), num_units=4,
                      stride=2),
      resnet_v1_block('block3', base_depth=depth_func(256), num_units=23,
                       stride=2),
      resnet_v1_block('block4', base_depth=depth_func(512), num_units=3,
                       stride=1),
  ]
  return resnet_v1(inputs, blocks, num_classes, is_training,
                    global_pool=global_pool, output_stride=output_stride,
                    include_root_block=True, spatial_squeeze=spatial_squeeze,
                    store_non_strided_activations=store_non_strided_activations,
                    reuse=reuse, scope=scope)
resnet_v1_101.default_image_size = resnet_v1.default_image_size
                       stride=2),
      resnet_v1_block('block4', base_depth=depth_func(512), num_units=3,
                       stride=1),
  ]
  return resnet_v1(inputs, blocks, num_classes, is_training,
                    global_pool=global_pool, output_stride=output_stride,
                    include_root_block=True, spatial_squeeze=spatial_squeeze,
                    store_non_strided_activations=store_non_strided_activations,
                    reuse=reuse, scope=scope)
```

Appendix D: Conclusion From the above techniques

From the SSD model introduced and the knowledge of the deep residual network, the researcher found that with the deepening of the depth convolution network level, the detection of objects on the training set will appear to decrease in accuracy. So in order to correct about defects from SSD model, the new algorithm proposed is better than the traditional SSD model in terms of detection accuracy. In view of the two defects of the SSD model above, the methods are introducing the two improves modules (Concat-sum module , Element-Sum module) using the deep VGG16 and the deep residual network with the feature pyramid network module are adopted, so the MSSD model is more accurate than the traditional detection accuracy. In image classification functions, outstanding networks have proven to be better than VGGS because it provides escape links between criminal blocks, thus gradually putting an end to hits and reducing the way the network is moving deeper. In fact, ResNets typically can go up to 101 layers whereas VGG networks can only go up to 16. Because deep nets are often better for sorting images, ResNets are generally more accurate than VGG. Here we will apply our idea on both of the VGG and ResNet networks showing the result that we got in our experiment above. The shallower layers (conv-3) suffering from lacking of sematic information, so in order to compensate that lack we inject contextual information from other layer and come out with new design model named element-sum and concat-sum model . Although replacing the VGG-based feature extractor in SSD with ResNet-101 does not lead to greater performance. Hence, a custom-made prediction module is needed.

APPENDIX E: Ethical Approval letter

NEAR EAST UNIVERSITY

ETHICAL APPROVAL DOCUMENT

Date: 20//11/2020

To the Graduate School of Applied Sciences

For the thesis project entitled as “ Multi-Layers Feature Fusion in SSD for Small Objects Detection”, the researchers declare that they did not collect any data from human/animal or any other subjects. Therefore, this project does not need to go through the ethics committee evaluation.

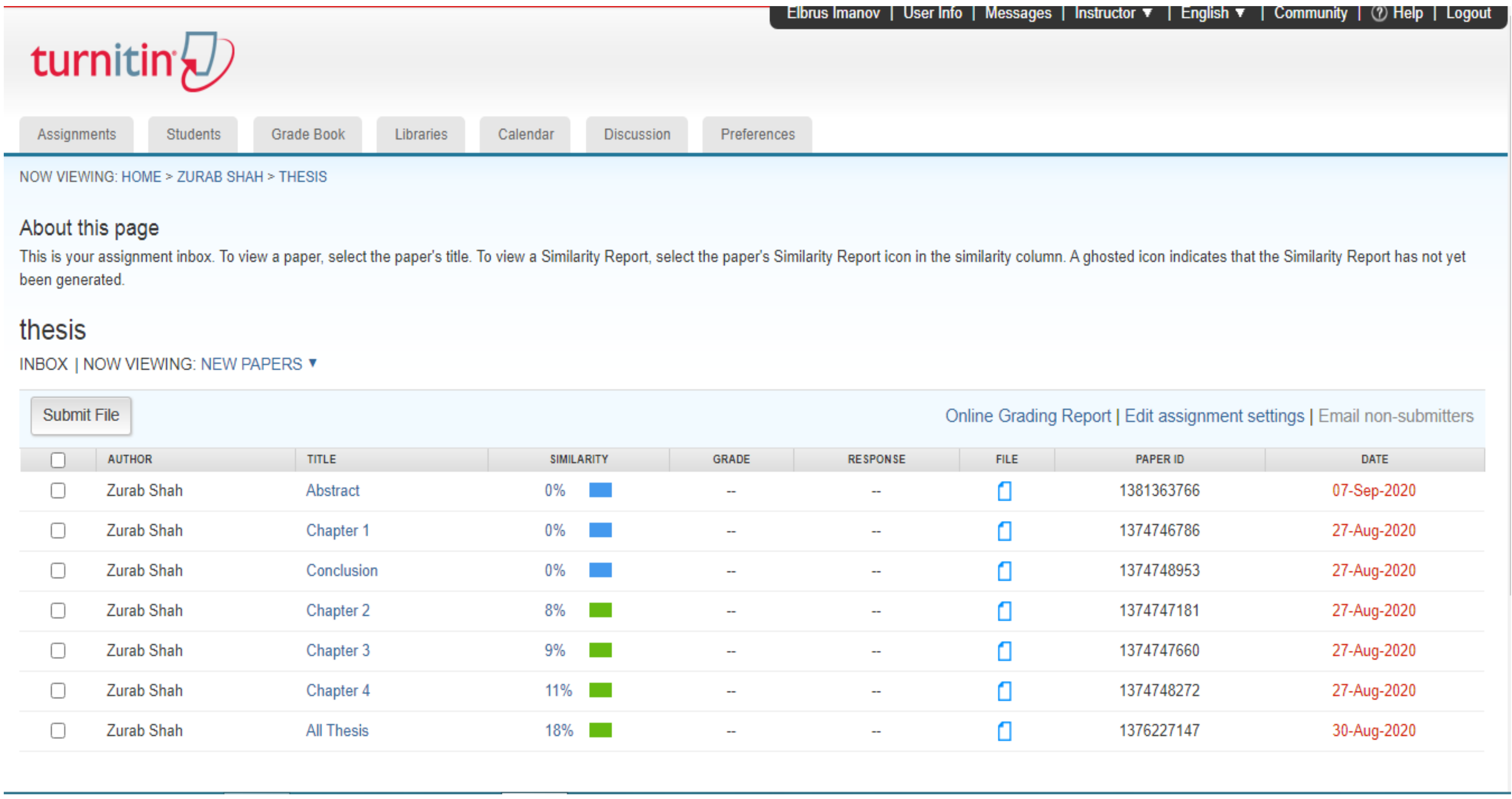
Title: Assist. Prof. Dr

Name Surname: Elbrus Imanov

Signature: [illegible]

Role in the Research Project: Supervisor

APPENDIX F: Turnitin Report



Elbrus Imanov | User Info | Messages | Instructor ▼ | English ▼ | Community | ? Help | Logout
turnitin
Assignments
Students
Grade Book
Libraries
Calendar
Discussion
Preferences

NOW VIEWING: HOME > ZURAB SHAH > THESIS

About this page

This is your assignment inbox. To view a paper, select the paper's title. To view a Similarity Report, select the paper's Similarity Report icon in the similarity column. A ghosted icon indicates that the Similarity Report has not yet been generated.

thesis

INBOX | NOW VIEWING: NEW PAPERS ▼

Submit File

Online Grading Report | Edit assignment settings | Email non-submitters

	AUTHOR	TITLE	SIMILARITY	GRADE	RESPONSE	FILE	PAPER ID	DATE
☐	Zurab Shah	Abstract	0% 🟦	--	--	▯	1381363766	07-Sep-2020
☐	Zurab Shah	Chapter 1	0% 🟦	--	--	▯	1374746786	27-Aug-2020
☐	Zurab Shah	Conclusion	0% 🟦	--	--	▯	1374748953	27-Aug-2020
☐	Zurab Shah	Chapter 2	8% 🟩	--	--	▯	1374747181	27-Aug-2020
☐	Zurab Shah	Chapter 3	9% 🟩	--	--	▯	1374747660	27-Aug-2020
☐	Zurab Shah	Chapter 4	11% 🟩	--	--	▯	1374748272	27-Aug-2020
☐	Zurab Shah	All Thesis	18% 🟩	--	--	▯	1376227147	30-Aug-2020

Title of Thesis: Multi-Layers Feature Fusion in SSD for Small Objects Detection

Assist. Prof. Dr. Elbrus Imanov [illegible]